C/C++ cheatsheet Documentation

Release 0.1.0

crazyguitar

Dec 26, 2022

CONTENTS

CHAPTER

ONE

C CHEAT SHEET

1.1 C Basic cheatsheet

Table of Contents

1.1.1 Comma Operator

```
#include <stdio.h>

#define PRINT(exp...)                           \
        {                                       \
                exp;                            \
                printf(#exp " => i = %d\n", i); \
        }
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{
        /* comma just a separators */
        int a = 1, b = 2, c = 3, i = 0;

        printf("(a, b, c) = (%d, %d, %d)\n\n", a, b, c);

        /* comma act as binary operator */
        PRINT( i = (a, b, c) );
        PRINT( i = (a + 5, a + b) );

        /* equivalent to (i = a + 5), a + b */
        PRINT( i = a + 5, a + b );

        return 0;
}
```

output:

```
$ ./a.out
(a, b, c) = (1, 2, 3)

i = (a, b, c) => i = 3
i = (a + 5, a + b) => i = 3
i = a + 5, a + b => i = 6
```

Note: Comma operator is a **binary operator**, it evaluates its first operand and discards the result, and then evaluates the second operand and return this value.

1.1.2 Old Style and New Style Function Definition

```
#include <stdio.h>

/* old style function declaration */
int old_style_add(a, b)
        int a; int b;
{
        return a + b;
}

/* new style function declaration */
int new_style_add(int a, int b)
{
        return a + b;
}

int main(int argc, char *argv[])
{
```

(continues on next page)

(continued from previous page)

```
        printf("old_sylte_add = %d\n", old_style_add(5566, 7788));
        printf("new_sylte_add = %d\n", new_style_add(5566, 9527));

        return 0;
}
```

output:

```
$ gcc -Wold-style-definition -g -Wall test.c
test.c: In function 'old_style_add':
test.c:4:5: warning: old-style function definition [-Wold-style-definition]
 int old_style_add(a, b)
     ^
$ ./a.out
old_sylte_add = 13354
new_sylte_add = 15093
```

1.1.3 sizeof(struct {int:-!!(e); }) Compile Time Assert

Reference

1. Stack Overflow
2. /usr/include/linux/kernel.h

```
#include <stdio.h>

#define FORCE_COMPILE_TIME_ERROR_OR_ZERO(e) \
        (sizeof(struct { int:-!!(e); }))

#define FORCE_COMPILE_TIME_ERROR_OR_NULL(e) \
        ((void *)sizeof(struct { int:-!!(e); }))

int main(int argc, char *argv[])
{
        FORCE_COMPILE_TIME_ERROR_OR_ZERO(0);
        FORCE_COMPILE_TIME_ERROR_OR_NULL(NULL);

        return 0;
}
```

output:

```
$ gcc test.c
$ tree .
.
|-- a.out
`-- test.c

0 directories, 2 files
```

```c
#include <stdio.h>

#define FORCE_COMPILE_TIME_ERROR_OR_ZERO(e) \
        (sizeof(struct { int:-!!(e); }))

#define FORCE_COMPILE_TIME_ERROR_OR_NULL(e) \
        ((void *)sizeof(struct { int:-!!(e); }))

int main(int argc, char *argv[])
{
        int a = 123;

        FORCE_COMPILE_TIME_ERROR_OR_ZERO(a);
        FORCE_COMPILE_TIME_ERROR_OR_NULL(&a);

        return 0;
}
```

output:

```
$ gcc test.c
test.c: In function 'main':
test.c:4:24: error: bit-field '<anonymous>' width not an integer constant
         (sizeof(struct { int:-!!(e); }))
                        ^
test.c:13:9: note: in expansion of macro 'FORCE_COMPILE_TIME_ERROR_OR_ZERO'
         FORCE_COMPILE_TIME_ERROR_OR_ZERO(a);
         ^
test.c:7:32: error: negative width in bit-field '<anonymous>'
         ((void *)sizeof(struct { int:-!!(e); }))
                                ^
test.c:14:9: note: in expansion of macro 'FORCE_COMPILE_TIME_ERROR_OR_NULL'
         FORCE_COMPILE_TIME_ERROR_OR_NULL(&a);
         ^
```

1.1.4 Machine endian check

```c
#include <stdio.h>
#include <stdint.h>

static union {
    uint8_t buf[2];
    uint16_t uint16;
} endian = { {0x00, 0x3a}};

#define LITTLE_ENDIAN ((char)endian.uint16 == 0x00)
#define BIG_ENDIAN ((char)endian.uint16 == 0x3a)



int main(int argc, char *argv[])
```

(continues on next page)

(continued from previous page)

```
{
    uint8_t buf[2] = {0x00, 0x3a};

    if (LITTLE_ENDIAN) {
        printf("Little Endian Machine: %x\n", ((uint16_t *)buf)[0]);
    } else {
        printf("Big Endian Machine: %x\n", ((uint16_t *)buf)[0]);
    }

    return 0;
}
```

output:

```
# on little endian macheine
$ ${CC} endian_check.c
$ ./a.out
Little Endian Machine: 3a00

# on big endian machine
$ ${CC} endian_check.c
$ ./a.out
Big Endian Machine: 3a
```

1.1.5 Implement closure via `static`

```
#include <stdio.h>

void foo()
{
    static int s_var = 9527;
    int l_var = 5566;

    l_var++;
    s_var++;
    printf("s_var = %d, l_var = %d\n", s_var, l_var);
}

int main(int argc, char *argv[])
{
    int i = 0;
    for (i=0; i < 5; i++) {
        foo();
    }
    return 0;
}
```

output:

```
$ ./a.out
s_var = 9528, l_var = 5567
```

(continues on next page)

(continued from previous page)

```
s_var = 9529, l_var = 5567
s_var = 9530, l_var = 5567
s_var = 9531, l_var = 5567
s_var = 9532, l_var = 5567
```

1.1.6 Split String

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

char ** split(char *str, const int sep)
{
    int num_cut = 1;
    int i = 0;
    char **buf = NULL;
    char *ptr = NULL;
    char delimiters[2] ={sep, '\0'};

    assert(str != NULL);
    printf("pattern = %s\n",str);
    for (ptr = str; *ptr != '\0'; ptr++) {
        if (*ptr == sep){ num_cut++; }
    }
    num_cut++;

    if (NULL == (buf = (char **)calloc(num_cut, sizeof(char *)))) {
        printf("malloc fail\n");
        goto Error;
    }

    ptr = strtok(str, delimiters);
    while (ptr != NULL) {
       buf[i++] = strdup(ptr);
       ptr = strtok(NULL, delimiters);
    }
Error:
    return buf;
}

void free_strlist(char **buf)
{
    char **ptr = NULL;
    for (ptr = buf; *ptr; ptr++) {
        free(*ptr);
    }
}

int main(int argc, char *argv[])
```

(continues on next page)

(continued from previous page)

```
{
    int ret = -1;
    char *pattern = NULL;
    char **buf = NULL;
    char **ptr = NULL;

    if (argc != 2) {
        printf("Usage: PROG string\n");
        goto Error;
    }

    pattern = argv[1];
    buf = split(pattern, ',');
    for (ptr = buf; *ptr; ptr++) {
        printf("%s\n",*ptr);
    }
    ret = 0;
Error:
    if (buf) {
        free_strlist(buf);
        buf = NULL;
    }
    return ret;
}
```

output:

```
$ ./a.out hello,world
pattern = hello,world
hello
world
```

1.1.7 Callback in C

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdint.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define CHECK_ERR(ret, fmt, ...)        \
    do {                                \
        if (ret < 0) {                  \
            printf(fmt, ##__VA_ARGS__); \
            goto End;                   \
        }                               \
    } while(0)
```

(continues on next page)

(continued from previous page)

```c
void callback(int err)
{
    if (err < 0) {
        printf("run task fail!\n");
    } else {
        printf("run task success!\n");
    }
}

int task(const char *path ,void (*cb)(int err))
{
    int ret = -1;
    struct stat st = {};

    ret = stat(path, &st);
    CHECK_ERR(ret, "stat(%s) fail. [%s]\n", path, strerror(errno));

    ret = 0;
End:
    cb(ret); /* run the callback function */
    return ret;
}


int main(int argc, char *argv[])
{
    int ret = -1;
    char *path = NULL;

    if (argc != 2) {
        printf("Usage: PROG [path]\n");
        goto End;
    }
    path = argv[1];
    task(path, callback);
    ret = 0;
End:
    return ret;
}
```

output:

```
$ ${CC} example_callback.c
$ ./a.out /etc/passwd
run task success!
$ ./a.out /etc/passw
stat(/etc/passw) fail. [No such file or directory]
run task fail!
```

1.1.8 Duff's device

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    int ret = -1, count = 0;
    int to = 0, from = 0;

    if (argc != 2) {
        printf("Usage: PROG [number]\n");
        goto End;
    }
    count = atoi(argv[1]);
    switch (count % 8) {
        case 0:        do {  to = from++;
        case 7:              to = from++;
        case 6:              to = from++;
        case 5:              to = from++;
        case 4:              to = from++;
        case 3:              to = from++;
        case 2:              to = from++;
        case 1:              to = from++;
                       } while ((count -= 8) > 0);
    }
    printf("get 'to': %d\n", to);
    ret = 0;
End:
    return ret;
}
```

output:

```
$ ./a.out 6
get 'to': 5
$ ./a.out
./test 19
get 'to': 18
```

1.1.9 `switch` goto `default` block

```c
#include <stdio.h>

enum { EVENT_FOO, EVENT_BAR, EVENT_BAZ, EVENT_QUX };

void demo(int event) {

    switch (event) {
        case EVENT_FOO:
            printf("---> foo event\n");
```

(continues on next page)

(continued from previous page)

```
            break;
        case EVENT_BAR:  while(1) {
                             printf("---> bar event\n");
                             break;
        case EVENT_BAZ:      printf("---> baz event\n");
                             break;
        case EVENT_QUX:      printf("---> qux event\n");
                             break;
                         }
        default:
            printf("default block\n");
    }
}

int main(int argc, char *argv[])
{
    demo(EVENT_FOO); /* will not fall into default block */
    demo(EVENT_BAR); /* will fall into default block */
    demo(EVENT_BAZ); /* will fall into default block */

    return 0;
}
```

output:

```
$ ./a.out
---> foo event
---> bar event
default block
---> baz event
default block
```

1.1.10 Simple `try ... catch` in C

```
/* cannot distinguish exception */

#include <stdio.h>
#include <setjmp.h>

enum {
    ERR_EPERM = 1,
    ERR_ENOENT,
    ERR_ESRCH,
    ERR_EINTR,
    ERR_EIO
};

#define try    do { jmp_buf jmp_env__;     \
                    if (!setjmp(jmp_env__))
#define catch       else
```

(continues on next page)

(continued from previous page)

```
#define end    } while(0)

#define throw(exc) longjmp(jmp_env__, exc)

int main(int argc, char *argv[])
{
    int ret = 0;

    try {
        throw(ERR_EPERM);
    } catch {
        printf("get exception!\n");
        ret = -1;
    } end;
    return ret;
}
```

output:

```
$ ./a.out
get exception!
```

1.1.11 Simple `try ... catch(exc)` in C

```
#include <stdio.h>
#include <string.h>
#include <setjmp.h>

enum {
    ERR_EPERM = 1,
    ERR_ENOENT,
    ERR_ESRCH,
    ERR_EINTR,
    ERR_EIO
};

#define try    do { jmp_buf jmp_env__;              \
                    switch ( setjmp(jmp_env__) ) { \
                        case 0:
#define catch(exc)          break;                 \
                        case exc:
#define end    } } while(0)

#define throw(exc) longjmp(jmp_env__, exc)

int main(int argc, char *argv[])
{
    int ret = 0;

    try {
```

(continues on next page)

(continued from previous page)

```
        throw(ERR_ENOENT);
    } catch(ERR_EPERM) {
        printf("get exception: %s\n", strerror(ERR_EPERM));
        ret = -1;
    } catch(ERR_ENOENT) {
        printf("get exception: %s\n", strerror(ERR_ENOENT));
        ret = -1;
    } catch(ERR_ESRCH) {
        printf("get exception: %s\n", strerror(ERR_ENOENT));
        ret = -1;
    } end;
    return ret;
}
```

output:

```
$ ./a.out
get exception: No such file or directory
```

1.1.12 Simple `try ... catch(exc) ... finally` in C

```
#include <stdio.h>
#include <string.h>
#include <setjmp.h>

enum {
    ERR_EPERM = 1,
    ERR_ENOENT,
    ERR_ESRCH,
    ERR_EINTR,
    ERR_EIO
};

#define try  do { jmp_buf jmp_env__  ;              \
                  switch ( setjmp(jmp_env__) ) { \
                      case 0: while(1) {
#define catch(exc)          break;                  \
                      case exc:
#define finally         break; }                    \
                  default:
#define end  } } while(0)

#define throw(exc) longjmp(jmp_env__, exc)

int main(int argc, char *argv[])
{
    int ret = 0;

    try {
        throw(ERR_ENOENT);
```

(continues on next page)

(continued from previous page)

```
    } catch(ERR_EPERM) {
        printf("get exception: %s\n", strerror(ERR_EPERM));
        ret = -1;
    } catch(ERR_ENOENT) {
        printf("get exception: %s\n", strerror(ERR_ENOENT));
        ret = -1;
    } catch(ERR_ESRCH) {
        printf("get exception: %s\n", strerror(ERR_ENOENT));
        ret = -1;
    } finally {
        printf("finally block\n");
    } end;
    return ret;
}
```

output:

```
$ ./a.out
get exception: No such file or directory
finally block
```

ref: Exceptions in C with Longjmp and Setjmp

1.1.13 Implement a Task Chain

```
#include <stdio.h>

typedef enum {
    TASK_FOO = 0,
    TASK_BAR,
    TASK_BAZ,
    TASK_NUM
} task_set;

#define NUM_TASKS TASK_NUM
#define LIST_ADD(list, ptr)       \
    do {                          \
        if (!list) {              \
            (list) = (ptr);       \
            ptr->prev = NULL;     \
            ptr->next = NULL;     \
        } else {                  \
            (list)->prev = ptr;   \
            (ptr)->next = (list); \
            (ptr)->prev = NULL;   \
            (list) = (ptr);       \
        }                         \
    } while(0)

struct task {
    task_set task_label;
```

(continues on next page)

(continued from previous page)

```
    void (*task) (void);
    struct task *next, *prev;
};

static void foo(void) { printf("Foo task\n"); }
static void bar(void) { printf("Bar task\n"); }
static void baz(void) { printf("Baz task\n"); }

struct task task_foo = { TASK_FOO, foo, NULL, NULL };
struct task task_bar = { TASK_BAR, bar, NULL, NULL };
struct task task_baz = { TASK_BAZ, baz, NULL, NULL };
static struct task *task_list = NULL;

static void register_task(struct task *t)
{
    LIST_ADD(task_list, t);
}

static void lazy_init(void)
{
    static init_done = 0;

    if (init_done == 0) {
        init_done = 1;

        /* register tasks */
        register_task(&task_foo);
        register_task(&task_bar);
        register_task(&task_baz);
    }
}

static void init_tasks(void) {
    lazy_init();
}

static struct task * get_task(task_set label)
{
    struct task *t = task_list;
    while (t) {
        if (t->task_label == label) {
            return t;
        }
        t = t->next;
    }
    return NULL;
}

#define RUN_TASK(label, ...)              \
    do {                                  \
        struct task *t = NULL;            \
        t = get_task(label);              \
```

(continues on next page)

(continued from previous page)

```
        if (t) { t-> task(__VA_ARGS__); } \
    } while(0)


int main(int argc, char *argv[])
{
    int i = 0;
    init_tasks();

    /* run chain of tasks */
    for (i=0; i<NUM_TASKS; i++) {
        RUN_TASK(i);
    }
    return 0;
}
```

output:

```
$ ./a.out
Foo task
Bar task
Baz task
```

1.2 GNU C Extensions cheatsheet

Table of Contents

- *structure & union initializer*
  - *Unnamed Structure and Union Fields*

1.2.1 with __extension__

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

/* with __extension__ */
#define lambda(ret_type, ...)                \
        __extension__                        \
        ({                                   \
                ret_type __fn__ __VA_ARGS__ \
                __fn__;                      \
        })

int main(int argc, char *argv[])
{
        int a = 5566, b = 9527;
        int c = __extension__ 0b101010;
        int (*max) (int, int) = lambda(int, (int x, int y) {return x > y ? x : y; });

        printf("max(%d, %d) = %d\n", a, b, max(a, b));
        printf("binary const c = %x\n", c);

        return 0;
}
#endif
```

output:

```
$ gcc -g -Wall -std=c99 -pedantic test.c
$ ./a.out
max(5566, 9527) = 9527
binary const c = 2a
```

1.2.2 without __extension__

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

/* with __extension__ */
#define lambda(ret_type, ...)                \
```

(continues on next page)

(continued from previous page)

```
        ({                                      \
                ret_type __fn__ __VA_ARGS__ \
                __fn__;                         \
        })

int main(int argc, char *argv[])
{
        int a = 5566, b = 9527;
        int c = 0b101010;
        int (*max) (int, int) = lambda(int, (int x, int y) {return x > y ? x : y; });

        printf("max(%d, %d) = %d\n", a, b, max(a, b));
        printf("binary const c = %x\n", c);

        return 0;
}
#endif
```

output:

```
$ gcc -g -Wall -pedantic test.c
test.c: In function 'main':
test.c:17:17: warning: binary constants are a GCC extension [enabled by default]
         int c = 0b101010;
                 ^
test.c:18:40: warning: ISO C forbids nested functions [-Wpedantic]
         int (*max) (int, int) = lambda(int, (int x, int y) {return x > y ? x : y; });
                                        ^
test.c:10:17: note: in definition of macro 'lambda'
                 ret_type __fn__ __VA_ARGS__ \
                 ^
test.c:9:9: warning: ISO C forbids braced-groups within expressions [-Wpedantic]
         ({                                      \
         ^
test.c:18:33: note: in expansion of macro 'lambda'
         int (*max) (int, int) = lambda(int, (int x, int y) {return x > y ? x : y; });
                                 ^
$ ./a.out
max(5566, 9527) = 9527
binary const c = 2a
```

Binary Constants

ref: Binary Constants

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{
        int a = 0b0101;
        int b = 0x003a;

        printf("%x, %x\n", a, b);

        return 0;
}
#endif
```

output:

```
$ gcc -g -Wall -pedantic test.c
test.c: In function 'main':
test.c:9:17: warning: binary constants are a GCC extension [enabled by default]
         int a = 0b0101;
                 ^
$ ./a.out
./a.out
5, 3a
```

Statements and Declarations in Expressions

ref: Statements and Declarations in Expressions

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define square(x)   \
  ({                \
        int y = 0;  \
        y = x * x;  \
        y;          \
   })

#define max(a, b)           \
  ({                        \
        typeof (a) _a = a;  \
        typeof (b) _b = b;  \
        _a > _b ? _a : _b;  \
   })

int main(int argc, char *argv[])
{
        int x = 3;
        int a = 55, b = 66;
        printf("square val: %d\n", square(x));
```

(continues on next page)

(continued from previous page)

```
        printf("max(%d, %d) = %d\n", a, b, max(a, b));
        return 0;
}

#endif
```

output:

```
$ ./a.out
square val: 9
max(55, 66) = 66
```

Locally Declared Labels

ref: Locally Declared Labels

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define ARRAYSIZE(arr)                          \
  ({                                            \
        size_t size = 0;                        \
        size = sizeof(arr) / sizeof(arr[0]);    \
        size;                                   \
   })

#define SEARCH(arr, size, target)           \
  ({                                        \
        __label__ found;                    \
        int i = 0;                          \
        int value = -1;                     \
        for (i = 0; i < size; i++) {        \
                if (arr[i] == target) {     \
                        value = i;          \
                        goto found;         \
                }                           \
        }                                   \
        value = -1;                         \
        found:                              \
        value;                              \
   })

int main(int argc, char *argv[])
{
        int arr[5] = {1, 2, 3, 9527, 5566};
        int target = 9527;

        printf("arr[%d] = %d\n",
```

(continues on next page)

(continued from previous page)

```
                SEARCH(arr, ARRAYSIZE(arr), target), target);
        return 0;
}

#endif
```

output:

```
$ ./a.out
arr[3] = 9527
```

Nested Functions

ref: Nested Functions

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        double a = 3.0;
        double square(double x) { return x * x; }

        printf("square(%.2lf) = %.2lf\n", a, square(a));
        return 0;
}
#endif
```

output:

```
$ ./a.out
square(3.00) = 9.00
```

Note: The nested function can access all the variables of the containing function that are visible at the point of its definition. This is called **lexical scoping**.

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        int i = 0;

        void up(void) { i++; }
```

(continues on next page)

(continued from previous page)

```
        printf("i = %d\n", i);
        up();
        printf("i = %d\n", i);
        up();
        printf("i = %d\n", i);

        return 0;
}
#endif
```

output:

```
./a.out
i = 0
i = 1
i = 2
```

Note: It is possible to call the nested function from outside the scope of its name by storing its address or passing the address to another function.

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define ARRAY_SIZE(arr) sizeof(arr) / sizeof(arr[0])
void print_str(char **arr, int i, char *(*access)(char **arr, int idx))
{
        char *ptr = NULL;

        if (arr == NULL) return;

        ptr = access(arr, i);
        if (ptr != NULL) {
                printf("str = %s\n", ptr);
        }
}

int main(int argc, char *argv[])
{
        char *arr[5] = {"Hello", "World", "Foo", "Bar", NULL};
        char *ptr = NULL;
        int i = 0;
        int offset = 1;

        char *access(char **arr, int idx)
        {
                return arr[idx + offset];
        }
```

(continues on next page)

(continued from previous page)

```
        for (i = 0; i < (ARRAY_SIZE(arr) - offset); i++) {
                print_str(arr, i, access);
        }

    return 0;
}
#endif
```

output:

```
$ ./a.out
str = World
str = Foo
str = Bar
```

Note: A nested function can jump to a label inherited from a containing function, provided the label is explicitly declared in the containing function.

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        __label__ end;
        int ret = -1, i = 0;

        void up(void)
        {
                i++;
                if (i > 2) goto end;
        }
        printf("i = %d\n", i); /* i = 0 */
        up();
        printf("i = %d\n", i); /* i = 1 */
        up();
        printf("i = %d\n", i); /* i = 2 */
        up();
        printf("i = %d\n", i); /* i = 3 */
        up();
        printf("i = %d\n", i); /* i = 4 */
        up();
        ret = 0;
end:
        return ret;
}
#endif
```

output:

```
$ ./a.out
i = 0
i = 1
i = 2
```

Note: If you need to declare the nested function before its definition, use `auto` (which is otherwise meaningless for function declarations).

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        int i = 0;
        auto void up(void);

        void up(void) { i++; }
        printf("i = %d\n", i); /* i = 0 */
        up();
        printf("i = %d\n", i); /* i = 1 */
        up();
        printf("i = %d\n", i); /* i = 2 */
        up();
        return 0;
}
#endif
```

output:

```
$ ./a.out
i = 0
i = 1
i = 2
```

Referring to a Type with `typeof`

ref: Referring to a Type with typeof

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define pointer(T)  typeof(T *)
#define array(T, N) typeof(T [N])
```

(continues on next page)

(continued from previous page)

```
int g_arr[5];

int main(int argc, char *argv[])
{
        int i = 0;
        char **ptr = NULL;

        /* This declares _val with the type of what ptr points to. */
        typeof (*g_arr) val = 5566;
        /* This declares _arr as an array of such values. */
        typeof (*g_arr) arr[3] = {1, 2,3};
        /* This declares y as an array of pointers to characters.*/
        array (pointer (char), 4) str_arr = {"foo", "bar", NULL};

        printf("val: %d\n", val);
        for (i = 0; i < 3; i++) {
                printf("arr[%d] = %d\n", i, arr[i]);
        }
        for (i = 0, ptr = str_arr; *ptr != NULL ; i++, ptr++) {
                printf("str_arr[%d] = %s\n", i, *ptr);
        }

        return 0;
}
#endif
```

output:

```
$ ./a.out
val: 5566
arr[0] = 1
arr[1] = 2
arr[2] = 3
str_arr[0] = foo
str_arr[1] = bar
```

Conditionals with Omitted Operands

ref: Conditionals with Omitted Operands

Note: The middle operand in a conditional expression may be omitted. Then if the first operand is nonzero, its value is the value of the conditional expression.

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{
        int x = 1, y = 0;
        int z = -1;

        /* equivalent to x ? x : y */
        z = x ? : y;
        printf("z = %d\n", z);
        return 0;
}
```

output:

```
$ ./a.out
z = 1
```

Arrays of Length Zero

ref: Zero-length arrays

Note: Zero-length arrays are allowed in GNU C. They are very useful as the **last element** of a structure which is really a header for a **variable-length** object

```
#include <stdlib.h>
#include <errno.h>
#include <string.h>

#define CHECK_NULL(ptr, fmt, ...)                   \
        do {                                        \
                if (!ptr) {                         \
                        printf(fmt, ##__VA_ARGS__); \
                        goto End;                   \
                }                                   \
        } while(0)

/* array item has zero length */
typedef struct _list {
        int len;
        char *item[0];
} list;

int main(int argc, char *argv[])
{

        int ret = -1, len = 3;
        list *p_list = NULL;

        p_list = (list *)malloc(sizeof(list) + sizeof(char *) * len);
        CHECK_NULL(p_list, "malloc fail. [%s]", strerror(errno));
```

(continues on next page)

(continued from previous page)

```
        p_list->item[0] = "Foo";
        p_list->item[1] = "Bar";
        p_list->item[2] = NULL;

        printf("item[0] = %s\n", p_list->item[0]);
        printf("item[1] = %s\n", p_list->item[1]);
        printf("item[2] = %s\n", p_list->item[2]);

        ret = 0;
End:

        if (p_list)
                free(p_list);

        return ret;
}

#endif
```

output:

```
$ ./a.out
item[0] = Foo
item[1] = Bar
item[2] = (null)
```

Note: GCC allows static initialization of flexible array members

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

typedef struct _list {
        int len;
        int item[];
} list;

#define PRINT_LIST(l)                                   \
        do {                                            \
                int i = 0;                              \
                for (i = 0; i < l.len; i++) {           \
                        printf("%d ", l.item[i]);       \
                }                                       \
                printf("\n");                           \
        } while(0)

int main(int argc, char *argv[])
{
```

(continues on next page)

(continued from previous page)

```
        static list l1 = {3, {1, 2, 3}};
        static list l2 = {5, {1, 2, 3, 4, 5}};

        PRINT_LIST(l1);
        PRINT_LIST(l2);
        return 0;
}

#endif
```

output:

```
$ ./a.out
1 2 3
1 2 3 4 5
```

Variadic Macros

ref: Variadic Macros

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define DEBUG_C99(fmt, ...)     fprintf(stderr, fmt, ##__VA_ARGS__)
#define DEBUG_GNU(fmt, args...) fprintf(stderr, fmt, ##args)

int main(int argc, char *argv[])
{
        DEBUG_C99("ISO C supported variadic macros\n");
        DEBUG_GNU("GNU C supported variadic macors\n");

        DEBUG_C99("ISO C format str = %s\n", "Foo");
        DEBUG_GNU("GNU C format str = %s\n", "Bar");

        return 0;
}
#endif
```

output:

```
$ ./a.out
ISO C supported variadic macros
GNU C supported variadic macors
ISO C format str = Foo
GNU C format str = Bar
```

Compound Literals (cast constructors)

ref: Compound Literals

Note: A compound literal looks like a cast containing an initializer. Its value is an object of the type specified in the cast, containing the elements specified in the initializer

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        struct foo {int a; char b[3]; } structure = {};

        /* compound literals (cast constructors )*/

        structure = ((struct foo) { 5566, 'a', 'b'});
        printf("a = %d, b = %s\n", structure.a, structure.b);

        /* equal to */

        struct foo temp = {5566, 'a', 'b'};
        structure = temp;

        printf("a = %d, b = %s\n", structure.a, structure.b);

        return 0;
}
#endif
```

output:

```
$ ./a.out
a = 5566, b = ab
a = 5566, b = ab
```

Note: If the object being initialized has array type of unknown size, the size is determined by compound literal size

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int main(int argc, char *argv[])
{
        /* The size is determined by compound literal size */
```

(continues on next page)

(continued from previous page)

```
        static int x[] = (int []) {1, 2, 3, 4, 5};
        static int y[] = (int [3]) {1};
        int i = 0;

        for (i = 0; i < 5; i++) printf("%d ", x[i]);
        printf("\n");

        for (i = 0; i < 3; i++) printf("%d ", y[i]);
        printf("\n");

        /* equal to */

        static int xx[] = {1, 2, 3, 4, 5};
        static int yy[] = {1, 0, 0};

        for (i = 0; i < 5; i++) printf("%d ", xx[i]);
        printf("\n");

        for (i = 0; i < 3; i++) printf("%d ", yy[i]);
        printf("\n");

        return 0;
}
#endif
```

output:

```
./a.out
1 2 3 4 5
1 0 0
1 2 3 4 5
1 0 0
```

Case Ranges

ref: Case Ranges

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

int foo(int a)
{
        switch (a) {
                case 1 ... 3:
                        return 5566;
                case 4 ... 6:
                        return 9527;
```

(continues on next page)

(continued from previous page)

```
        }
        return 7788;
}

int main(int argc, char *argv[])
{
        int b = 0;

        b = foo(1);
        printf("b = %d\n", b);

        b = foo(5);
        printf("b = %d\n", b);

        b = foo(10);
        printf("b = %d\n", b);

        return 0;
}
#endif
```

output:

```
$ ./a.out
b = 5566
b = 9527
b = 7788
```

Warning: Be careful, write spaces around the `...` (ex: `r1 ... r2`), for otherwise it may be parsed wrong when you use it with integer values

Designated Initializers

ref: Initializers

1.2.3 Array initializer

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define ARRLEN 6

int main(int argc, char *argv[])
{
        /* ISO C99 support giving the elements in any order */
```

(continues on next page)

(continued from previous page)

```
        int a[ARRLEN] = {[5] = 5566, [2] = 9527};
        /* equal to (ISO C90)*/
        int b[ARRLEN] = {0, 0, 9527, 0, 0, 5566};
        register int i = 0;

        for (i = 0; i < ARRLEN; i++) printf("%d ", a[i]);
        printf("\n");

        for (i = 0; i < ARRLEN; i++) printf("%d ", a[i]);
        printf("\n");

        return 0;
}
#endif
```

output:

```
$ # compile in C90 mode
$ gcc -std=c90 -pedantic test.c
test.c: In function 'main':
test.c:12:26: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         int a[ARRLEN] = {[5] = 5566, [2] = 9527};
                          ^
test.c:12:38: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         int a[ARRLEN] = {[5] = 5566, [2] = 9527};
                                      ^

$ # compile in C99 mode
$ gcc -std=c99 -pedantic test.c
$ ./a.out
0 0 9527 0 0 5566
0 0 9527 0 0 5566
```

Note: GNU C also support to initialize a range of elements to the same value

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

#define ARRLEN 10

int main(int argc, char *argv[])
{
        int arr[ARRLEN] = { [2 ... 5] = 5566, [7 ... 9] = 9527};
        register i = 0;

        for (i = 0; i< ARRLEN; i++) printf("%d ", arr[i]);
        printf("\n");
```

(continues on next page)

(continued from previous page)

```
        return 0;
}
#endif
```

output:

```
$ gcc -pedantic test.c
test.c: In function 'main':
test.c:11:32: warning: ISO C forbids specifying range of elements to initialize [-
↪Wpedantic]
         int arr[ARRLEN] = { [2 ... 5] = 5566, [7 ... 9] = 9527};
                                ^
test.c:11:29: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         int arr[ARRLEN] = { [2 ... 5] = 5566, [7 ... 9] = 9527};
                             ^
test.c:11:50: warning: ISO C forbids specifying range of elements to initialize [-
↪Wpedantic]
         int arr[ARRLEN] = { [2 ... 5] = 5566, [7 ... 9] = 9527};
                                                  ^
test.c:11:47: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         int arr[ARRLEN] = { [2 ... 5] = 5566, [7 ... 9] = 9527};
                                               ^
$ ./a.out
0 0 5566 5566 5566 5566 0 9527 9527 9527
```

1.2.4 structure & union initializer

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

typedef struct _point {int x, y; } point;
typedef union _foo {int i; double d; } foo;


int main(int argc, char *argv[])
{
        point a = { 5566, 9527 };
        /* GNU C support initialize with .fieldname = */
        point b = { .x = 5566, .y = 9527 };
        /* obsolete since GCC 2.5 */
        point c = { x: 5566, y: 9527 };
        /* specify which element of the union should be used */
        foo bar = { .d = 5566 };

        printf("a.x = %d, a.y = %d\n", a.x, a.y);
        printf("b.x = %d, b.y = %d\n", b.x, b.y);
```

(continues on next page)

(continued from previous page)

```
        printf("c.x = %d, c.y = %d\n", c.x, c.y);
        printf("bar.d = %.2lf\n", bar.d);

        return 0;
}
#endif
```

output:

```
$ gcc -pedantic test.c
test.c: In function 'main':
test.c:15:21: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         point b = { .x = 5566, .y = 9527 };
                     ^
test.c:15:32: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         point b = { .x = 5566, .y = 9527 };
                                ^
test.c:17:22: warning: obsolete use of designated initializer with ':' [-Wpedantic]
         point c = { x: 5566, y: 9527 };
                      ^
test.c:17:31: warning: obsolete use of designated initializer with ':' [-Wpedantic]
         point c = { x: 5566, y: 9527 };
                                ^
test.c:19:21: warning: ISO C90 forbids specifying subobject to initialize [-Wpedantic]
         foo bar = { .d = 5566 };
                     ^
test.c:24:9: warning: ISO C90 does not support the '%lf' gnu_printf format [-Wformat=]
         printf("bar.d = %.2lf\n", bar.d);
         ^
$ a.out
a.x = 5566, a.y = 9527
b.x = 5566, b.y = 9527
c.x = 5566, c.y = 9527
bar.d = 5566.00
```

Unnamed Structure and Union Fields

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

struct foo {
        int a;
        union {
                int b;
                char byte[4];
        };
        int d;
};
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{

        struct foo bar = { 0x1a, { 0x2b }, 0x3c };
        int i = 0;

        printf("%x, %x, %x\n", bar.a, bar.b, bar.d);

        /* on little machine, we will get 2b 0 0 0 */
        for (i = 0; i < 4; i++) printf("%x ", bar.byte[i]);
        printf("\n");

        return 0;
}
#endif
```

output:

```
# without gcc options -std=c11 will raise warning
$ gcc -g -Wall -pedantic test.c
test.c:12:10: warning: ISO C90 doesn't support unnamed structs/unions [-Wpedantic]
         };
          ^
# with gcc options -std=c11 will not raise warning
$ gcc -g -Wall -pedantic -std=c11 test.c
$ ./a.out
1a, 2b, 3c
2b 0 0 0
```

Note: Unnamed field must be a structure or union definition without a tag like `struct { int a; };`. If `-fms-extensions` is used, the field may also be a definition with a tag such as `struct foo { int a; };`

```
#ifndef __GNUC__
#error "__GNUC__ not defined"
#else

#include <stdio.h>

struct foo {
        int b;
        int c;
};

struct bar {
        int a;
        struct foo;
        int d;
};
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{
        struct bar baz = { 0x1a, { 0x2b, 0x00 }, 0x3c };

        printf("%x, %x, %x, %x\n", baz.a, baz.b, baz.c, baz.d);

        return 0;
}
#endif
```

output:

```
$ gcc -g -Wall -pedantic -std=c11 -fms-extensions test.c
$ ./a.out
1a, 2b, 0, 3c
```

1.3 C Macro cheatsheet

Table of Contents

1.3.1 Predefined Macros

Macro	descriptions
__FILE__	current file name
__DATE__	current compile date in “MMM DD YYYY”” format.
__TIME__	current compile time in “HH:MM:SS” format.
__LINE__	current line number
__func__	current function name

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int ret = -1;

    printf("__FILE__: %s\n"
           "__DATE__: %s\n"
           "__TIME__: %s\n"
           "__LINE__: %d\n"
           "__func__: %s\n",
           __FILE__, __DATE__, __TIME__, __LINE__, __func__);

    ret = 0;
    return ret;
}
```

output:

```
$ cc -g -Wall -o test test.c
$ ./test
__FILE__: test.c
__DATE__: Sep 28 2016
__TIME__: 10:01:59
__LINE__: 16
__func__: main
```

1.3.2 DEBUG switch

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int ret = -1;

#ifdef DEBUG
    printf("debug version\n");
#else
    printf("release version\n");
#endif

    ret = 0;
    return ret;
}
```

output:

```
$ cc -g -Wall -o test test.c
$ ./test
release version
$ cc -g -Wall -DDEBUG -o test test.c
```

(continues on next page)

(continued from previous page)

```
$ ./test
debug version
```

1.3.3 ARRAYSIZE

```
#include <stdio.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))

/*
 * Entry point
 */
int main(int argc, char *argv[])
{
    int ret = -1;
    char *pszArr[] = {"Hello", "World", NULL};

    printf("array size: %lu\n", ARRAY_SIZE(pszArr));
    ret = 0;
    return ret;
}
```

output:

```
$ cc -g -Wall -o test test.c
$ ./test
array size: 3
```

1.3.4 FOREACH

```
#include <stdio.h>

#define FOREACH(item, arr) \
    for (item=arr; *item; item++)

/*
 * Entry point
 */
int main(int argc, char *argv[])
{
    int ret = -1;
    char *pszArr[] = {"Hello", "World", NULL};
    char **str = NULL;

    FOREACH (str, pszArr) {
        printf("%s ", *str);
    }
    printf("\n");
```

(continues on next page)

(continued from previous page)

```
    ret = 0;
    return ret;
}
```

output:

```
$ cc -g -Wall -o test test.c
$ ./test
Hello World
```

1.3.5 ALLOC_STRUCT

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#define ALLOC_STRUCT(s)  ((s *) malloc(sizeof(s)))
#define EXPECT_NOT_NULL(i, ...) \
    if (i == NULL) { __VA_ARGS__ }
#define EXPECT_ALLOC_SUCCESS(i, fmt, ...) \
    EXPECT_NOT_NULL(i, printf(fmt, ##__VA_ARGS__); goto End;)

typedef struct _foo {
    int hello;
    int world;
} foo;

int main(int argc, char *argv[])
{
    int ret = -1;
    foo *f  = NULL;
    f = ALLOC_STRUCT(foo);
    EXPECT_ALLOC_SUCCESS(f, "err: %s", strerror(errno));
    printf("alloc foo success\n");
    ret = 0;
End:
    return ret;
}
```

output:

```
$ gcc -g -Wall -o test test.c
$ ./test
alloc foo success
```

1.3.6 lambda

```
#define lambda(return_type, ...) \
    __extension__ \
    ({ \
        return_type __fn__ __VA_ARGS__ \
        __fn__; \
    })

/*
 * Entry point
 */
int main(int argc, char *argv[])
{
    int ret = -1;
    int (*max) (int, int) =
        lambda (int, (int x, int y) { return x > y ? x : y; });

    printf("lambda: %d\n", max(2,3));

    ret = 0;
    return ret;
}
```

output:

```
$ gcc -g -Wall -o test test.c
$ ./test
lambda: 3
```

1.3.7 EXPECT_*

```
#include <stdio.h>
↪                                                                    [19/1840]
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define EXPECT_TRUE(i, ...) \
    if (i != 1) { __VA_ARGS__ }

#define EXPECT_FALSE(i, ...) \
    if (i != 0) { __VA_ARGS__ }

#define EXPECT_EQ(i, e, ...) \
    if (i != e) { __VA_ARGS__ }

#define EXPECT_NEQ(i, e, ...) \
    if (i == e) { __VA_ARGS__ }
```

(continues on next page)

(continued from previous page)

```
#define EXPECT_LT(i, e, ...) \
    if (i >= e) { __VA_ARGS__ }

#define EXPECT_LE(i, e, ...) \
    if (i > e) { __VA_ARGS__ }

#define EXPECT_GT(i, e, ...) \
    if (i <= e) { __VA_ARGS__ }

#define EXPECT_GE(i, e, ...) \
    if (i < e) { __VA_ARGS__ }

#define EXPECT_SUCCESS(ret, fmt, ...) \
    EXPECT_GT(ret, 0, \
        printf(fmt, ##__VA_ARGS__); \
        goto End; \
    )

/*
 * Entry point
 */
int main(int argc, char *argv[])
{
    int ret = -1;

    EXPECT_TRUE(1);
    EXPECT_FALSE(0);
    EXPECT_LT(1, 0, printf("check less then fail\n"););
    EXPECT_GT(0, 1, printf("check great then fail\n"););
    EXPECT_SUCCESS(ret, "ret = %d\n", ret);
    ret = 0;
End:
    return ret;
}
```

output:

```
$ cc -g -Wall -o checkerr checkerr.c
$ ./checkerr
check less then fail
check great then fail
ret = -1
```

1.3.8 Get struct member *GET_FIELD_PTR*

```c
#include <stdio.h>

#define _GET_FIELD_OFFSET(s, field ) \
    ((short)(long)(&((s *)NULL)->field))

#define _GET_FIELD_PTR(ps, offset) \
    ((void *)(((char *)ps) + (offset)))

#define GET_FIELD_PTR(s, ps, field) \
    _GET_FIELD_PTR(ps, _GET_FIELD_OFFSET(s, field))

typedef struct _foo {
    char name[16];
    int age;
    int gender;
} foo;

/*
 * Entry point
 */
int main(int argc, char *argv[])
{
    int ret = -1;
    char *name = NULL;
    int *age = NULL, *gender = NULL;
    foo f = {.name="c", .age=44, .gender=0};

    name   = GET_FIELD_PTR(foo, &f, name);
    age    = GET_FIELD_PTR(foo, &f, age);
    gender = GET_FIELD_PTR(foo, &f, gender);

    printf("name: %s\n"
           "age: %d\n"
           "gender: %d\n", name, *age, *gender);

    ret = 0;
    return ret;
}
```

output:

```
$ cc -g -Wall -o test test.c
$ ./test
name: c
age: 44
gender: 0
```

1.3.9 define __attribute__ ((*))

```
#if __GNUC__ >= 3
#undef inline
#define inline          inline __attribute__ ((always_inline))
#define __noinline       __attribute__ ((noinline))
#define __pure           __attribute__ ((pure))
#define __const          __attribute__ ((const))
#define __noreturn       __attribute__ ((noreturn))
#define __malloc         __attribute__ ((malloc))
#define __must_check     __attribute__ ((warn_unused_result))
#define __deprecated     __attribute__ ((deprecated))
#define __used           __attribute__ ((used))
#define __unused         __attribute__ ((unused))
#define __packed         __attribute__ ((packed))
#define __align(x)       __attribute__ ((aligned, (x)))
#define __align_max      __attribute__ ((aligned))
#define likely(x)        __builtin_expect (!!(x), 1)
#define unlikely(x)      __builtin_expect (!!(x), 0)
#else
#undef inline
#define __noinline   /* no noinline           */
#define __pure       /* no pure               */
#define __const      /* no const              */
#define __noreturn   /* no noreturn           */
#define __malloc     /* no malloc             */
#define __must_check /* no warn_unused_result */
#define __deprecated /* no deprecated         */
#define __used       /* no used               */
#define __unused     /* no unused             */
#define __packed     /* no packed             */
#define __align(x)   /* no aligned            */
#define __align_max  /* no align_max          */
#define likely(x)    (x)
#define unlikely(x)  (x)
#endif
```

1.4 C Makefile cheatsheet

Table of Contents

- *build executable files respectively*
- *using `$(eval)` predefine variables*
- *build subdir and link together*
- *build shared library*
- *build shared and static library*
- *build recursively*
- *replace current shell*
- *one line condition*
- *Using define to control CFLAGS*

1.4.1 Automatic variables

automatic variables	descriptions
$@	The file name of the target
$<	The name of the first prerequisite
$^	The names of all the prerequisites
$+	prerequisites listed more than once are duplicated in the order

Makefile

```
.PHONY: all

all: hello world

hello world: foo foo foo bar bar
        @echo "== target: $@ =="
        @echo $<
        @echo $^
        @echo $+

foo:
        @echo "Hello foo"

bar:
        @echo "Hello Bar"
```

output

```
Hello foo
Hello Bar
== target: hello ==
foo
foo bar
foo foo foo bar bar
== target: world ==
foo
```

(continues on next page)

(continued from previous page)

```
foo bar
foo foo foo bar bar
```

1.4.2 using $(warning text) check make rules (for debug)

```
$(warning Top level warning)

FOO := $(warning FOO variable)foo
BAR  = $(warning BAR variable)bar

$(warning target)target: $(warning prerequisite list)Makefile $(BAR)
        $(warning tagrget script)
        @ls
$(BAR):
```

output

```
Makefile:1: Top level warning
Makefile:3: FOO variable
Makefile:6: target
Makefile:6: prerequisite list
Makefile:6: BAR variable
Makefile:9: BAR variable
Makefile:7: tagrget script
Makefile
```

1.4.3 string functions

Makefile

```
SRC      = hello_foo.c hello_bar.c foo_world.c bar_world.c

SUBST    = $(subst .c,,$(SRC))

SRCST    = $(SRC:.c=.o)
PATSRCST = $(SRC:%.c=%.o)
PATSUBST = $(patsubst %.c, %.o, $(SRC))

.PHONY: all

all: sub filter findstring words word wordlist

sub:
        @echo "== sub example =="
        @echo "SUBST: " $(SUBST)
        @echo "SRCST: " $(SRCST)
        @echo "PATSRCST: " $(PATSRCST)
        @echo "PATSUBST: " $(PATSUBST)
        @echo ""
```

(continues on next page)

(continued from previous page)

```
filter:
        @echo "== filter example =="
        @echo "filter: " $(filter hello_%, $(SRC))
        @echo "filter-out: $(filter-out hello_%, $(SRC))"
        @echo ""

findstring:
        @echo "== findstring example =="
        @echo "Res: " $(findstring hello, hello world)
        @echo "Res: " $(findstring hello, ker)
        @echo "Res: " $(findstring world, worl)
        @echo ""

words:
        @echo "== words example =="
        @echo "num of words: "$(words $(SRC))
        @echo ""


word:
        @echo "== word example =="
        @echo "1st word: " $(word 1,$(SRC))
        @echo "2nd word: " $(word 2,$(SRC))
        @echo "3th word: " $(word 3,$(SRC))
        @echo ""


wordlist:
        @echo "== wordlist example =="
        @echo "[1:3]:"$(wordlist 1,3,$(SRC))
        @echo ""
```

output

```
$ make
== sub example ==
SUBST:  hello_foo hello_bar foo_world bar_world
SRCST:  hello_foo.o hello_bar.o foo_world.o bar_world.o
PATSRCST:  hello_foo.o hello_bar.o foo_world.o bar_world.o
PATSUBST:  hello_foo.o hello_bar.o foo_world.o bar_world.o

== filter example ==
filter:  hello_foo.c hello_bar.c
filter-out: foo_world.c bar_world.c

== findstring example ==
Res:  hello
Res:
Res:

== words example ==
```

(continues on next page)

(continued from previous page)

```
num of words: 4

== word example ==
1st word:  hello_foo.c
2nd word:  hello_bar.c
3th word:  foo_world.c

== wordlist example ==
[1:3]:hello_foo.c hello_bar.c foo_world.c
```

1.4.4 using `$(sort list)` sort list and remove duplicates

Makefile

```
SRC = foo.c bar.c ker.c foo.h bar.h ker.h

.PHONY: all

all:
        @echo $(suffix $(SRC))
        @echo $(sort $(suffix $(SRC)))
```

output

```
$ make
.c .c .c .h .h .h
.c .h
```

1.4.5 single dollar sign and double dollar sign

dollar sign	descriptions
$	reference a make variable using $
$$	reference a shell variable using $$

Makefile

```
LIST = one two three

.PHONY: all single_dollar double_dollar

all: single_dollar double_dollar

double_dollar:
        @echo "=== double dollar sign example ==="
        @for i in $(LIST); do \
                echo $$i;     \
        done

single_dollar:
```

(continues on next page)

(continued from previous page)

```
        @echo "=== single dollar sign example ==="
        @for i in $(LIST); do  \
                echo $i;     \
        done
```

output

```
$ make
=== single dollar sign example ===



=== double dollar sign example ===
one
two
three
```

1.4.6 build executable files respectively

directory layout

```
.
|-- Makefile
|-- bar.c
|-- bar.h
|-- foo.c
`-- foo.h
```

Makefile

```
# CFLAGS: Extra flags to give to the C compiler
CFLAGS   += -Werror -Wall -O2 -g
SRC       = $(wildcard *.c)
OBJ       = $(SRC:.c=.o)
EXE       = $(subst .c,,$(SRC))

.PHONY: all clean

all: $(OBJ) $(EXE)

clean:
    rm -rf *.o *.so *.a *.la $(EXE)
```

output

```
$ make
cc -Werror -Wall -O2 -g   -c -o foo.o foo.c
cc -Werror -Wall -O2 -g   -c -o bar.o bar.c
cc   foo.o   -o foo
cc   bar.o   -o bar
```

1.4.7 using $(eval) predefine variables

without $(eval)

```
SRC = $(wildcard *.c)
EXE = $(subst .c,,$(SRC))

define PROGRAM_template
$1_SHARED = lib$(strip $1).so
endef

.PHONY: all

$(foreach exe, $(EXE), $(call PROGRAM_template, $(exe)))

all:
        @echo $(foo_SHARED)
        @echo $(bar_SHARED)
```

output

```
$ make
Makefile:11: *** missing separator.  Stop.
```

with $(evall)

```
CFLAGS  += -Wall -g -O2 -I./include
SRC = $(wildcard *.c)
EXE = $(subst .c,,$(SRC))

define PROGRAM_template
$1_SHARED = lib$(strip $1).so
endef

.PHONY: all

$(foreach exe, $(EXE), $(eval $(call PROGRAM_template, $(exe))))

all:
        @echo $(foo_SHARED)
        @echo $(bar_SHARED)
```

output

```
$ make
libfoo.so
libbar.so
```

1.4.8 build subdir and link together

directory layout

```
.
|-- Makefile
|-- include
|   `-- foo.h
`-- src
    |-- foo.c
    `-- main.c
```

Makefile

```
CFLAGS  += -Wall -g -O2 -I./include
SRC     = $(wildcard src/*.c)
OBJ     = $(SRC:.c=.o)
EXE     = main

.PHONY: all clean

all: $(OBJ) $(EXE)

$(EXE): $(OBJ)
        $(CC) $(LDFLAGS) -o $@ $^

%.o: %.c
        $(CC) $(CFLAGS) -c $< -o $@

clean:
        rm -rf *.o *.so *.a *.la $(EXE) src/*.o src/*.so src/*a
```

output

```
$ make
cc -Wall -g -O2 -I./include -c src/foo.c -o src/foo.o
cc -Wall -g -O2 -I./include -c src/main.c -o src/main.o
cc  -o main src/foo.o src/main.o
```

1.4.9 build shared library

directory layout

```
.
|-- Makefile
|-- include
|   `-- common.h
`-- src
    |-- bar.c
    `-- foo.c
```

Makefile

```
SONAME    = libfoobar.so.1
SHARED    = src/libfoobar.so.1.0.0
SRC       = $(wildcard src/*.c)
OBJ       = $(SRC:.c=.o)

CFLAGS    += -Wall -Werror -fPIC -O2 -g -I./include
LDFLAGS   += -shared -Wl,-soname,$(SONAME)

.PHONY: all clean

all: $(SHARED) $(OBJ)

$(SHARED): $(OBJ)
        $(CC) $(LDFLAGS) -o $@ $^

%.o: %.c
        $(CC) $(CFLAGS) -c $^ -o $@

clean:
        rm -rf src/*.o src/*.so.* src/*.a src/*.la
```

output

```
$ make
cc -Wall -Werror -fPIC -O2 -g -I./include -c src/foo.c -o src/foo.o
cc -Wall -Werror -fPIC -O2 -g -I./include -c src/bar.c -o src/bar.o
cc -shared -Wl,-soname,libfoobar.so.1 -o src/libfoobar.so.1.0.0 src/foo.o src/bar.o
```

1.4.10 build shared and static library

directory layout

```
.
|-- Makefile
|-- include
|   |-- bar.h
|   `-- foo.h
`-- src
    |-- Makefile
    |-- bar.c
    `-- foo.c
```

Makefile

```
SUBDIR = src

.PHONY: all clean $(SUBDIR)

all: $(SUBDIR)

clean: $(SUBDIR)
```

(continues on next page)

(continued from previous page)

```
$(SUBDIR):
        make -C $@ $(MAKECMDGOALS)
```

src/Makefile

```
SRC      = $(wildcard *.c)
OBJ      = $(SRC:.c=.o)
LIB      = libfoobar

STATIC   = $(LIB).a
SHARED   = $(LIB).so.1.0.0
SONAME   = $(LIB).so.1
SOFILE   = $(LIB).so

CFLAGS  += -Wall -Werror -g -O2 -fPIC -I../include
LDFLAGS += -shared -Wl,-soname,$(SONAME)

.PHONY: all clean

all: $(STATIC) $(SHARED) $(SONAME) $(SOFILE)

$(SOFILE): $(SHARED)
        ln -sf $(SHARED) $(SOFILE)

$(SONAME): $(SHARED)
        ln -sf $(SHARED) $(SONAME)

$(SHARED): $(STATIC)
        $(CC) $(LDFLAGS) -o $@ $<

$(STATIC): $(OBJ)
        $(AR) $(ARFLAGS) $@ $^

%.o: %.c
        $(CC) $(CFLAGS) -c -o $@ $<

clean:
        rm -rf *.o *.a *.so *.so.*
```

output

```
$ make
make -C src
make[1]: Entering directory '/root/test/src'
cc -Wall -Werror -g -O2 -fPIC -I../include -c -o foo.o foo.c
cc -Wall -Werror -g -O2 -fPIC -I../include -c -o bar.o bar.c
ar rv libfoobar.a foo.o bar.o
ar: creating libfoobar.a
a - foo.o
a - bar.o
cc -shared -Wl,-soname,libfoobar.so.1 -o libfoobar.so.1.0.0 libfoobar.a
ln -sf libfoobar.so.1.0.0 libfoobar.so.1
```

(continues on next page)

(continued from previous page)

```
ln -sf libfoobar.so.1.0.0 libfoobar.so
make[1]: Leaving directory '/root/test/src'
```

1.4.11 build recursively

directory layout

```
.
|-- Makefile
|-- include
|   `-- common.h
|-- src
|   |-- Makefile
|   |-- bar.c
|   `-- foo.c
`-- test
    |-- Makefile
    `-- test.c
```

Makefile

```
SUBDIR = src test

.PHONY: all clean $(SUBDIR)

all: $(SUBDIR)

clean: $(SUBDIR)

$(SUBDIR):
        $(MAKE) -C $@ $(MAKECMDGOALS)
```

src/Makefile

```
SONAME   = libfoobar.so.1
SHARED   = libfoobar.so.1.0.0
SOFILE   = libfoobar.so

CFLAGS  += -Wall -g -O2 -Werror -fPIC -I../include
LDFLAGS += -shared -Wl,-soname,$(SONAME)

SRC      = $(wildcard *.c)
OBJ      = $(SRC:.c=.o)

.PHONY: all clean

all: $(SHARED) $(OBJ)

$(SHARED): $(OBJ)
        $(CC) $(LDFLAGS) -o $@ $^
        ln -sf $(SHARED) $(SONAME)
```

(continues on next page)

(continued from previous page)

```
        ln -sf $(SHARED) $(SOFILE)

%.o: %.c
        $(CC) $(CFLAGS) -c $< -o $@

clean:
        rm -rf *.o *.so.* *.a *.so
```

test/Makefile

```
CFLAGS    += -Wall -Werror -g -I../include
LDFLAGS   += -Wall -L../src -lfoobar

SRC        = $(wildcard *.c)
OBJ        = $(SRC:.c=.o)
EXE        = test_main

.PHONY: all clean

all: $(OBJ) $(EXE)

$(EXE): $(OBJ)
        $(CC) -o $@ $^ $(LDFLAGS)

%.o: %.c
        $(CC) $(CFLAGS) -c $< -o $@

clean:
        rm -rf *.so *.o *.a $(EXE)
```

output

```
$ make
make -C src
make[1]: Entering directory '/root/proj/src'
cc -Wall -g -O2 -Werror -fPIC -I../include -c foo.c -o foo.o
cc -Wall -g -O2 -Werror -fPIC -I../include -c bar.c -o bar.o
cc -shared -Wl,-soname,libfoobar.so.1 -o libfoobar.so.1.0.0 foo.o bar.o
ln -sf libfoobar.so.1.0.0 libfoobar.so.1
ln -sf libfoobar.so.1.0.0 libfoobar.so
make[1]: Leaving directory '/root/proj/src'
make -C test
make[1]: Entering directory '/root/proj/test'
cc -Wall -Werror -g -I../include -c test.c -o test.o
cc -o test_main test.o -Wall -L../src -lfoobar
make[1]: Leaving directory '/root/proj/test'
$ tree .
.
|-- Makefile
|-- include
|   `-- common.h
|-- src
```

(continues on next page)

(continued from previous page)

```
|   |-- Makefile
|   |-- bar.c
|   |-- bar.o
|   |-- foo.c
|   |-- foo.o
|   |-- libfoobar.so -> libfoobar.so.1.0.0
|   |-- libfoobar.so.1 -> libfoobar.so.1.0.0
|   `-- libfoobar.so.1.0.0
`-- test
    |-- Makefile
    |-- test.c
    |-- test.o
    `-- test_main

3 directories, 14 files
```

1.4.12 replace current shell

```
OLD_SHELL := $(SHELL)
SHELL = /usr/bin/python

.PHONY: all

all:
        @import os; print os.uname()[0]
```

output

```
$ make
Linux
```

1.4.13 one line condition

syntax: `$(if cond, then part, else part)`

Makefile

```
VAR =
IS_EMPTY = $(if $(VAR), $(info not empty), $(info empty))

.PHONY: all

all:
        @echo $(IS_EMPTY)
```

output

```
$ make
empty
```

(continues on next page)

(continued from previous page)

```
$ make VAR=true
not empty
```

1.4.14 Using define to control CFLAGS

Makefile

```
CFLAGS += -Wall -Werror -g -O2
SRC     = $(wildcard *.c)
OBJ     = $(SRC:.c=.o)
EXE     = $(subst .c,,$(SRC))

ifdef DEBUG
CFLAGS += -DDEBUG
endif

.PHONY: all clean

all: $(OBJ) $(EXE)

clean:
        rm -rf $(OBJ) $(EXE)
```

output

```
$ make
cc -Wall -Werror -g -O2   -c -o foo.o foo.c
cc   foo.o   -o foo
$ make DEBUG=1
cc -Wall -Werror -g -O2 -DDEBUG   -c -o foo.o foo.c
cc   foo.o   -o foo
```

1.5 X86 Assembly cheatsheet

Table of Contents

1.5.1 Exit

```
# gcc -o a.out -nostdlib  a.s

.global _start
.section .text

_start:

mov $0x1,%eax   # 32 bit of exit is 1
mov $0x1,%ebx
int $0x80

.section .data
```

Note that `int 0x80` always invokes 32-bit system calls. To use system calls define on X64 systems, we need to use `syscall` instruction.

```
.global _start
.section .text

_start:

mov $0x3c,%eax   # 64 bit of exit is 60(0x3c)
mov $0x1,%ebx
syscall

.section .data
```

1.5.2 Hello Word

```
# gcc -o a.out -nostdlib  a.s
# ./a.out
# Hello World

.global _start
.section .text

_start:

# write(stdout, "Hello World", 13);

mov $0x4,%eax       # 32 bit write syscall number
mov $0x1,%ebx       # unsigned int fd (stdout)
lea (message),%ecx  # const char *buf
mov $13,%edx        # size_t count
int $0x80

# exit(0)

mov $0x1,%eax
```

(continues on next page)

(continued from previous page)

```
mov $0x0,%ebx
int $0x80

.section .data
message:
.ascii "Hello World\n"
```

1.5.3 do while

```
.global _start
.section .text

_start:

mov $0x1,%rsi

loop: # do {

# write(stdout, "Hello World\n", 13)
mov $0x4,%eax
mov $0x1,%ebx
lea (message),%ecx
mov $13,%edx
int $0x80

add $0x1,%rsi
cmp $0x5,%rsi
jbe loop  # } while(i<=5)

# exit
mov $0x1,%eax
mov $0x0,%ebx
int $0x80

.section .data
message: .ascii "Hello World\n"
```

1.5.4 Procedures

```
.global _start
.section .text

_start:

callq print

# exit
mov $0x1,%eax
mov $0x0,%ebx
```

(continues on next page)

(continued from previous page)

```
int $0x80

print:
# write(stdout, "Hello World\n", 13)
mov $0x4,%eax
mov $0x1,%ebx
lea (message),%ecx
mov $13,%edx
int $0x80
ret

.section .data
message: .ascii "Hello World\n"
```

1.5.5 Reference

- Linux System Call Table
- x86_64 Assembly Linux System Call Confusion

CHAPTER

TWO

MODERN C++ CHEAT SHEET

2.1 Basic cheatsheet

Table of Contents

2.1.1 C Linkage

```
#include <iostream>

#ifdef __cplusplus
extern "C" {
#endif

int fib(int n) {
  int a = 0, b = 1;
  for (int i = 0; i < n; ++i) {
    auto x = b;
    b = a + b;
    a = x;
  }
```

(continues on next page)

(continued from previous page)

```
  return a;
}

#ifdef __cplusplus
}
#endif

int main(int argc, char *argv[]) {
  std::cout << fib(10) << "\n";
}
// $ g++ -std=c++17 -Wall -Werror -O3 a.cc
// $ nm -g a.out | grep fib
// 0000000100003a58 T _fib
```

```
#include <iostream>

int fib(int n) {
  int a = 0, b = 1;
  for (int i = 0; i < n; ++i) {
    auto x = b;
    b = a + b;
    a = x;
  }
  return a;
}

int main(int argc, char *argv[]) {
  std::cout << fib(10) << "\n";
}
// $ g++ -std=c++17 -Wall -Werror -O3 a.cc
// nm -g a.out | grep fib
// 0000000100003a58 T __Z3fibi
```

```
#include <iostream>
#include <sys/cdefs.h>

__BEGIN_DECLS

int fib(int n) {
  int a = 0, b = 1;
  for (int i = 0; i < n; ++i) {
    auto x = b;
    b = a + b;
    a = x;
  }
  return a;
}

__END_DECLS

int main(int argc, char *argv[]) {
```

(continues on next page)

(continued from previous page)

```
  std::cout << fib(10) << "\n";
}
// $ g++ -std=c++17 -Wall -Werror -O3 a.cc
// $ nm -g a.out | grep fib
// 0000000100003a58 T _fib
```

2.1.2 Uniform Initialization

Uniform Initialization is also called braced initialization, which unifies constructing an object using a brace. However, there are some pitfalls in using syntax. For example, the compiler prefers to call `std::initializer_list` to initialize an object even with a matched constructor. The following snippet shows that `x{10, 5.0}` will call `Foo(std::initializer_list<long double>)` to construct an object event though `Foo(int a, double b)` is the more suitable one.

```
#include <iostream>
#include <initializer_list>

class Foo {
public:
  Foo(int a, double b) {
    std::cout << "without initializer_list\n";
  }

  Foo(std::initializer_list<long double> il) {
    std::cout << "with initializer_list\n";
  }
};

int main(int argc, char *argv[]) {
  Foo x{10, 5.0};
   // output: with initializer_list
}
```

Moreover, *uniform initialization* does not support narrowing conversion. Therefore, the following snippet will compile errors because `int` and `double` need to do narrowing conversion `bool`.

```
#include <iostream>
#include <initializer_list>

class Foo {
public:
  Foo(int a, double b) {
    std::cout << "without initializer_list\n";
  }

  // compile error
  Foo(std::initializer_list<bool> il) {
    std::cout << "with initializer_list\n";
  }
};
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
  Foo x{10, 5.0};
}
```

Note that when types cannot convert, the compiler does not use `std::initializer_list` to initialize an object. For example, `int` and `double` cannot convert to `std::string`, so the compiler will call `Foo(int, double)` to create an object.

```
#include <iostream>
#include <string>
#include <initializer_list>

class Foo {
public:
  Foo(int a, double b) {
    std::cout << "without initializer_list\n";
  }

  Foo(std::initializer_list<std::string> il) {
    std::cout << "with initializer_list\n";
  }
};

int main(int argc, char *argv[]) {
  Foo x{10, 5.0};
  // output: without initializer_list
}
```

2.1.3 Negative Array index

```
#include <iostream>

int main(int argc, char *argv[]) {
    // note: arr[i] = *(a + i)
    int arr[] = {1, 2, 3};
    int *ptr = &arr[1];

    std::cout << ptr[-1] << "\n";
    std::cout << ptr[0] << "\n";
    std::cout << ptr[1] << "\n";
}
```

2.1.4 Reference

```
#include <iostream>

template<typename T>
void f(T& param) noexcept {}
// param is a reference

int main(int argc, char *argv[])
{
    int x = 123;
    const int cx = x;
    const int &rx = x;

    f(x);   // type(param) = int&
    f(cx);  // type(param) = const int&
    f(rx);  // type(param) = const int&

    return 0;
}
```

```
#include <iostream>

template<typename T>
void f(T&& param) noexcept {}
// param is a universal reference

int main(int argc, char *argv[])
{
    int x = 123;
    const int cx = x;
    const int &rx = x;

    f(x);   // x is a lvalue, type(param) = int&
    f(cx);  // cx is a lvalue, type(param) = const int&
    f(rx);  // rx is a lvalue, type(param) = const int&
    f(12);  // 12 is a rvalue, type(param) = int&&

    return 0;
}
```

```
#include <iostream>

template<typename T>
void f(T param) noexcept {}
// param is neither a pointer nor a reference.

int main(int argc, char *argv[])
{
    int x = 123;
    const int cx = x;
    const int &rx = x;
```

(continues on next page)

(continued from previous page)

```
    f(x);   // type(param) = int
    f(cx);  // type(param) = int
    f(rx);  // type(param) = int
    f(12);  // type(param) = int

    return 0;
}
```

2.1.5 auto

```
auto x = 123;         // type(x) = int
const auto cx = x;    // type(cx) = const int
const auto &rx = x;   // type(rx) = const int&

auto &&urx = x;       // type(urx) = int&
auto &&urcx = cx;     // type(urcx) = const int&
auto &&urrx = rx;     // type(urrx) = const int&
auto &&urrv = 12;     // type(urrv) = int&&
```

2.1.6 decltype(auto)

The `decltype(auto)` is similar to auto, which decudes type via compiler. However, `decltype(auto)` preserves types reference and cv-qualifiers, while auto does not.

```
#include <type_traits>

int main(int argc, char *argv[]) {
  int x;
  const int cx = x;
  const int &crx = x;
  int &&z = 0;

  // decltype(auto) preserve cv-qualifiers
  decltype(auto) y1 = crx;
  static_assert(std::is_same<const int &, decltype(y1)>::value == 1);
  // auto does not preserve cv-qualifiers
  auto y2 = crx;
  static_assert(std::is_same<int, decltype(y2)>::value == 1);
  // decltype(auto) preserve rvalue reference
  decltype(auto) z1 = std::move(z);
  static_assert(std::is_same<int &&, decltype(z1)>::value == 1);
}
```

`decltype(auto)` is especially useful for writing a generic function's return.

```
#include <type_traits>

auto foo(const int &x) {
```

(continues on next page)

(continued from previous page)

```
  return x;
}

decltype(auto) bar(const int &x) {
  return x;
}

int main(int argc, char *argv[]) {
  static_assert(std::is_same<int, decltype(foo(1))>::value == 1);
  static_assert(std::is_same<const int &, decltype(bar(1))>::value == 1);
}
```

2.1.7 Reference Collapsing

```
// T& & -> T&
// T& && -> T&
// T&& & -> T&
// T&& && -> T&&
// note & always wins. that is T& && == T&& & == T& & == T&
// only T&& && == T&&
```

2.1.8 Perfect Forwarding

```
#include <iostream>
#include <utility>
#include <type_traits>

template <typename T>
T&& forward(typename std::remove_reference<T>::type& t) noexcept {
  std::cout << std::is_lvalue_reference<decltype(t)>::value << std::endl;
  return static_cast<T&&>(t);
}

template <typename T>
T&& forward(typename std::remove_reference<T>::type&& t) noexcept {
  static_assert(
    !std::is_lvalue_reference<T>::value,
    "Can not forward an rvalue as an lvalue."
  );
  std::cout << std::is_lvalue_reference<decltype(t)>::value << std::endl;
  return static_cast<T&&>(t);
}

int main (int argc, char *argv[])
{
  int a = 0;
  forward<int>(a);     // forward lvalues to rvalues
  forward<int>(9527);  // forward rvalues to rvalues
```

(continues on next page)

(continued from previous page)

```
  return 0;
}
```

```
#include <iostream>
#include <utility>
#include <type_traits>

template <typename T, typename Func>
void wrapper(T &&a, Func fn) {
  fn(std::forward<T>(a)); // forward lvalue to lvalues or rvalues
}

struct Foo {
  Foo(int a1, int a2) : a(a1), b(a2), ret(0) {}
  int a, b, ret;
};

int main (int argc, char *argv[])
{
  Foo foo{1, 2};
  Foo &bar = foo;
  Foo &&baz = Foo(5, 6);

  wrapper(foo, [](Foo foo) {
    foo.ret =  foo.a + foo.b;
    return foo.ret;
  });
  std::cout << foo.ret << std::endl;

  wrapper(bar, [](Foo &foo) {
    foo.ret = foo.a - foo.b;
    return foo.ret;
  });
  std::cout << bar.ret << std::endl;

  // move an rvalue to lvalue
  wrapper(std::move(baz), [](Foo &&foo) {
    foo.ret = foo.a * foo.b;
    return foo.ret;
  });
  std::cout << baz.ret << std::endl;
  return 0;
}
```

2.1.9 Bit Manipulation

```
#include <iostream>
#include <bitset>

int main(int argc, char *argv[]) {
        std::bitset<4> b{8};

        // show number of bits set
        std::cout << b.count() << "\n";
        // compare with int
        std::cout << (b == 8) << "\n";
}
```

2.1.10 Using `std::addressof`

Because C++ allows the overloading of `operator &`, accessing the address of an reference will result in infinite recursion. Therefore, when it is necessary to access the address of reference, it would be safer by using `std::addressof`.

```
#include <iostream>
#include <memory>

struct A {
  int x;
};

const A *operator &(const A& a) {
  // return &a; <- infinite recursion
  return std::addressof(a);
}

int main(int argc, char *argv[]) {
  A a;
  std::cout << &a << "\n";
}
```

2.2 constructor

Table of Contents

2.2.1 Constructors

```cpp
#include <iostream>
#include <utility>

class C {
 public:
  // constructor
  C(int x) : x_(x) {}

  // default constructor
  C() = default;

  // copy constructor
  C(const C &other) : C(other.x_) {
    std::cout << "copy constructor\n";
  }

  // copy assignment
  C &operator=(const C &other) {
    std::cout << "copy assignment\n";
    x_ = other.x_;
    return *this;
  }

  // move constructor
  C(C &&other) : x_(std::move(other.x_)) {
    std::cout << "move constructor\n";
    other.x_ = 0;
  }

  // move assignment
  C &operator=(C &&other) {
    std::cout << "move assignment\n";
    x_ = std::move(other.x_);
    return *this;
  }

 private:
  int x_;
};

int main(int argc, char *argv[]) {
  C c1;                   // call default constructor
  C c2(1);                // call constructor
  C c3 = C(2);            // call constructor
  C c4(c2);               // call copy constructor
  C c5(std::move(C(2)));  // call move constructor
  C c6 = c1;              // call copy constructor
  C c7 = std::move(C(2)); // call move constructor
  C c8 = std::move(c3);   // call move constructor

  C c9;
```

(continues on next page)

(continued from previous page)

```
  C c10;

  c9 = c2;                // call copy assignment
  c10 = std::move(c4);    // call move assignment
  c10 = C(2);             // call move assignment
}
```

2.2.2 Rule of three

```
#include <iostream>
#include <memory>
#include <string>
#include <cstring>

class RuleOfThree {
 public:
  RuleOfThree(const char *s, size_t n)
    : cstr_(new char[n])
    , n_(n) {
     memcpy(cstr_, s, n);
  }

  // if we have a user-defined destructor
  ~RuleOfThree() { delete[] cstr_; }
  // we need one a user-defined copy constructor
  RuleOfThree(const RuleOfThree &other)
    : RuleOfThree(other.cstr_, other.n_) {}
  // and user-defined copy assignment
  RuleOfThree &operator=(const RuleOfThree &other) {
    if (this == std::addressof(other)) {
      return *this;
    }
    delete[] cstr_;
    n_ = other.n_;
    cstr_ = new char[other.n_];
    memcpy(cstr_, other.cstr_, n_);
    return *this;
  }

  friend std::ostream &operator<<(std::ostream &os, const RuleOfThree &);

 private:
  char *cstr_;
  size_t n_;
};

std::ostream &operator<<(std::ostream &os, const RuleOfThree &r) {
  return os << r.cstr_;
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
  std::string s = "Rule of three";
  RuleOfThree r3(s.c_str(), s.size() + 1);
  std::cout << r3 << "\n";
}
```

2.2.3 Rule of five

```
#include <iostream>
#include <memory>
#include <string>
#include <cstring>
#include <utility>

class RuleOfFive {
 public:
  RuleOfFive(const char *s, int n) : cstr_(new char[n]) {
    std::memcpy(cstr_, s, n);
  }

  // if there is a user-defined destructor including default or delete
  ~RuleOfFive() { delete[] cstr_; }
  // a user-defined copy constructor
  RuleOfFive(const RuleOfFive &other)
    : RuleOfFive(other.cstr_, strlen(other.cstr_) + 1) {}
  // a user-defined move constructor
  RuleOfFive(RuleOfFive &&other)
    : cstr_(std::exchange(other.cstr_, nullptr)) {}
  // a user-define copy assignment
  RuleOfFive &operator=(const RuleOfFive &other) {
    return *this = RuleOfFive(other);
  }
  // a user-defined move assignment have to declare explicitly.
  RuleOfFive &operator=(RuleOfFive &&other) {
    std::swap(cstr_, other.cstr_);
    return *this;
  }

  friend std::ostream &operator<<(std::ostream &os, const RuleOfFive &);

 private:
  char *cstr_;
};

std::ostream &operator<<(std::ostream &os, const RuleOfFive &r5) {
  return os << r5.cstr_;
}

int main(int argc, char *argv[]) {
  std::string s = "Rule of five";
```

(continues on next page)

(continued from previous page)

```
  RuleOfFive r5(s.c_str(), s.size() + 1);
  std::cout << r5 << "\n";
}
```

2.2.4 Rule of zero

```
#include <iostream>
#include <string>

class RuleOfZero {
 public:
  RuleOfZero(const std::string &s) : s_(s) {}
  // if we don't have a user-defined destructor, we should not have
  // user-defined copy/move constructors or copy/move assignment.
  friend std::ostream &operator<<(std::ostream &os, const RuleOfZero &r0);
 private:
  const std::string s_;
};

std::ostream &operator<<(std::ostream &os, const RuleOfZero &r0) {
  return os << r0.s_;
}

int main(int argc, char *argv[]) {
  RuleOfZero r0("Rule of zero");
  std::cout << r0 << "\n";
}
```

Note that a polymorphic class should supress public copy/move.

```
#include <iostream>
#include <string>
#include <utility>

// bad
class A {
 public:
  virtual std::string f() { return "a"; }
};

class B : public A {
 public:
  std::string f() override { return "b"; }
};

void func(A &a) {
  auto c = a;
  std::cout << c.f() << "\n";
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
  B b;
  func(b);
}
```

```
#include <iostream>
#include <string>
#include <utility>

class A {
 public:
  A() = default;
  A(const A&) = delete;
  A &operator=(const A&) = delete;
  virtual std::string f() { return "a"; }
};

class B : public A {
 public:
  std::string f() override { return "b"; }
};

void func(A &a) {
  auto c = a;  // compile error here!
  std::cout << c.f() << "\n";
}

int main(int argc, char *argv[]) {
  B b;
  func(b);
}
```

2.3 Initialization

Table of Contents

2.3.1 Initializer lists

```
#include <iostream>
#include <initializer_list>

template<typename T>
decltype(auto) sum(const std::initializer_list<T> &v) {
    T s = 0;
    for (const auto &i : v) {
        s += i;
    }
    return s;
}

int main(int argc, char *argv[]) {
    sum<int>({1,2,3,4,5});
}
```

2.4 String

Table of Contents

2.4.1 Char to a string

```
#include <string>

int main(int argc, char *argv[]) {
  // string(size_t n, char c)
  std::string a(1, 'a');
}
```

```
#include <string>

int main(int argc, char *argv[]) {
  std::string s;
```

(continues on next page)

(continued from previous page)

```
  s += 'a';
}
```

```
#include <string>

int main(int argc, char *argv[]) {
  std::string s;
  s = 'a';
}
```

2.4.2 C String to a String

```
#include <string>

int main(int argc, char *argv[]) {
  char cstr[] = "hello cstr";
  std::string s = cstr;
}
```

2.4.3 Split a String

```
// $ g++ --std=c++14 -Wall -Werror -g -O3 split.cpp
// $ ./a.out
// abc
// def
// ghi

#include <iostream>
#include <string>
#include <vector>

using namespace std;

vector<string> split(const string &str, char delimeter) {

    string s = str;
    vector<string> out;
    size_t pos = 0;

    while((pos = s.find(delimeter)) != string::npos) {
        string token = s.substr(0, pos);
        out.emplace_back(token);
        s.erase(0, pos + 1);
    }
    out.emplace_back(s);
    return out;
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {

    string s = "abc,def,ghi";
    vector<string> v = split(s, ',');
    for (const auto &c : v) {
        cout << c << "\n";
    }
}
```

Using istream

```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

using namespace std;

template<char delimiter>
class String : public string
{
    friend istream &operator>>( istream  &is, String &out) {
        std::getline(is, out, delimiter);
        return is;
    }
};

int main(int argc, char *argv[]) {
    std::string text = "abc,def,ghi";

    istringstream iss(text);
    vector<string> out((istream_iterator<String<','>>(iss)),
                       istream_iterator<String<','>>());

    for (const auto &c : out) {
        cout << c << "\n";
    }
}
```

Using `std::getline`

```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

using namespace std;

int main(int argc, char *argv[])
{
    string in = "abc,def,ghi";
    vector<string> out;
```

(continues on next page)

(continued from previous page)

```
    string token;
    std::istringstream stream(in);

    while (std::getline(stream, token, ',')) {
        out.emplace_back(token);
    }
    for (const auto &c : out) {
        cout << c << "\n";
    }
}
```

Using boost

```
#include <iostream>
#include <string>
#include <vector>
#include <boost/algorithm/string.hpp>

using namespace std;

int main(int argc, char *argv[]) {
    string in = "abc,def,ghi";
    vector<string> out;

    boost::split(out, in, [](char c) { return c == ','; });
    for (const auto &s : out) {
        cout << s << "\n";
    }
}
```

2.4.4 Upper & Lower

```
// cc -std=c++17 -Wall -Werror -O3 a.cpp

#include <iostream>
#include <string>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::string s = "Hello World";
  // to upper
  std::transform(s.begin(), s.end(), s.begin(), ::toupper);
  std::cout << s << "\n";

  // to lower
  std::transform(s.begin(), s.end(), s.begin(), ::tolower);
  std::cout << s << "\n";
}
```

2.4.5 String Concat

Note that concatenating a string at the beginning is much slower than appending in the end. Although reserving space can speed up inserting a string in front of another one, the performance is still much slower than appending a string at the back.

```
    #include <iostream>
    #include <chrono>

    constexpr int total = 100000;
    using milliseconds = std::chrono::milliseconds;

    template <typename F>
    void profile(F &&func) {
      const auto start = std::chrono::steady_clock::now();
      func();
      const auto end = std::chrono::steady_clock::now();
      const auto d = end - start;
      const auto mill = std::chrono::duration_cast<milliseconds>(d).count();
      std::cout << mill << " ms\n";
    }

    int main(int argc, char *argv[]) {

      profile([] {
            std::string s;
            for (int i = 0; i < total; ++i) {
              s += 'a';
            }
      });

      profile([] {
            std::string s;
            for (int i = 0; i < total; ++i) {
              s = std::string(1, 'a') + s;
            }
      });

      profile([] {
        std::string s;
        s.reserve(total+1);
        for (int i = 0; i < total; ++i) {
          s = std::string(1, 'a') + s;
        }
      });
    }

// $ g++ -std=c++17 -Wall -Werror a.cc
// 0 ms
// 143 ms
// 110 ms
```

2.4.6 String Literals

```
#include <iostream>
#include <string>
#include <string_view>

int main(int argc, char *argv[]) {
  using namespace std::literals;

  auto s1 = "c string";
  auto s2 = "std::string"s;
  auto s3 = "std::string_view"sv;

  std::cout << s1 << "\n";
  std::cout << s2 << "\n";
  std::cout << s3 << "\n";
}
```

2.4.7 String View

```
#include <iostream>
#include <string_view>

void f(std::string_view s) {
  std::cout << s << "\n";
}

int main(int argc, char *argv[]) {
  const std::string s = "foo";
  // pass a const string is ok
  f(s);
}
```

```
#include <iostream>
#include <string_view>

void f(std::string s) {
  std::cout << s << "\n";
}

int main(int argc, char *argv[]) {
  std::string_view s = "foo";
  f(s); // compile error. cannot convert a string_view to a string
}
```

```
#include <iostream>
#include <string_view>

void f(std::string s) {
  std::cout << s << "\n";
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
  std::string_view s = "foo";
  // we can cast a string_view to a string
  f(static_cast<std::string>(s));
}
```

```
// string_view is not alway has null-terminated
#include <iostream>
#include <cstring>
#include <string_view>

int main(int argc, char *argv[]) {
  char array[3] = {'B', 'a', 'r'};
  std::string_view s(array, sizeof array);
  // Dangerous!! ptr will access memory address larger than array+3
  for (auto ptr = s.data(); !!ptr; ++ptr) {
    std::cout << *ptr << "\n";
  }
}
```

2.5 Container

2.5.1 Priority Queue

Table of Contents

2.5.2 Priority Queue

```
#include <iostream>
#include <functional>
#include <vector>
#include <queue>

template<typename Q>
void dump(Q &q) {
  while(!q.empty()) {
    std::cout << q.top() << " ";
    q.pop();
  }
  std::cout << "\n";
}

void foo() {
  std::vector<int> data{1, 5, 2, 1, 3};
  std::priority_queue<int> queue;
  for (auto & x : data) { queue.push(x); }
  dump(queue);
}

void bar() {
  std::vector<int> data{1, 5, 2, 1, 3};
  std::priority_queue<int, std::vector<int>, std::greater<int>> queue;
  for (auto & x : data) { queue.push(x); }
  dump(queue);
}

void baz() {
  std::vector<int> data{1, 5, 2, 1, 3};
  auto cmp = [](int x, int y) { return x < y; };
  std::priority_queue<int, std::vector<int>, decltype(cmp)> queue(cmp);
  for (auto & x : data) { queue.push(x); }
  dump(queue);
}

int main(int argc, char *argv[]) {
  foo();
  bar();
  baz();
  // 5 3 2 1 1
  // 1 1 2 3 5
  // 1 1 2 3 5
}
```

Priority queue is useful when a programmer need to merge multiple lists of data in order.

```
#include <iostream>
#include <vector>
#include <queue>
```

(continues on next page)

(continued from previous page)

```
template<typename Q>
void dump(Q &q) {
  while(!q.empty()) {
    std::cout << q.top() << " ";
    q.pop();
  }
  std::cout << "\n";
}

int main(int argc, char *argv[]) {
  std::priority_queue<int> queue;
  std::vector<int> x{9, 7, 8};
  std::vector<int> y{0, 5, 3};
  for (auto &e : x) { queue.push(e); }
  for (auto &e : y) { queue.push(e); }
  dump(queue);
  // 9 8 7 5 3 0
}
```

2.5.3 Profiling

```
// profile.h
#include <iostream>
#include <chrono>

using milliseconds = std::chrono::milliseconds;

template <typename T, typename F>
void profile(T &t, F &func) {
  const auto start = std::chrono::steady_clock::now();
  func(t);
  const auto end = std::chrono::steady_clock::now();
  const auto d = end - start;
  const auto mill = std::chrono::duration_cast<milliseconds>(d).count();
  std::cout << mill << " ms\n";
}
```

Push Front

```
// g++ -O3 -std=c++17 -Wall -Werror -I${HDR} a.cpp

#include <vector>
#include <deque>
#include <list>
#include <range/v3/view/iota.hpp>
#include "profile.h"

template <typename T>
void insert(T &t) {
```

(continues on next page)

(continued from previous page)

```
  for (auto i : ranges::views::iota(0, 300000)) {
    t.insert(t.begin(), i);
  }
}

int main(int argc, char *argv[]) {
  std::vector<int> v;
  std::deque<int> q;
  std::list<int> l;
  profile(v, insert<decltype(v)>);
  profile(q, insert<decltype(q)>);
  profile(l, insert<decltype(l)>);
}
```

```
$ ./a.out
16045 ms
1 ms
6 ms
```

Push Back

```
#include <vector>
#include <deque>
#include <list>
#include <range/v3/view/iota.hpp>
#include "profile.h"

template <typename T>
void insert(T &t) {
  for (auto i : ranges::views::iota(0, 1000000)) {
    t.push_back(i);
  }
}

int main(int argc, char *argv[]) {
  std::vector<int> v;
  std::deque<int> q;
  std::list<int> l;
  profile(v, insert<decltype(v)>);
  profile(q, insert<decltype(q)>);
  profile(l, insert<decltype(l)>);
}
```

```
./a.out
7 ms
2 ms
39 ms
```

Pop Front

```cpp
#include <vector>
#include <deque>
#include <list>
#include <range/v3/view/iota.hpp>
#include "profile.h"

template <typename T>
void insert(T &t) {
  for (auto i : ranges::views::iota(0, 300000)) {
    t.push_back(i);
  }
}

template <typename T>
void pop_front(T &t) {
  while (!t.empty()) {
    t.pop_front();
  }
}

template <typename T>
void erase(T &v) {
  while(!v.empty()) {
    v.erase(v.begin());
  }
}

int main(int argc, char *argv[]) {
  std::vector<int> v;
  std::deque<int> q;
  std::list<int> l;
  insert(v); insert(q); insert(l);
  profile(v, erase<decltype(v)>);
  profile(q, pop_front<decltype(q)>);
  profile(l, pop_front<decltype(l)>);
}
```

```
$ ./a.out
22923 ms
0 ms
12 ms
```

Pop Back

```cpp
#include <vector>
#include <deque>
#include <list>
#include <range/v3/view/iota.hpp>
#include "profile.h"

template <typename T>
void insert(T &t) {
  for (auto i : ranges::views::iota(0, 1000000)) {
    t.push_back(i);
  }
}

template <typename T>
void pop_back(T &t) {
  while (!t.empty()) {
    t.pop_back();
  }
}

int main(int argc, char *argv[]) {
  std::vector<int> v;
  std::deque<int> q;
  std::list<int> l;
  insert(v); insert(q); insert(l);
  profile(v, pop_back<decltype(v)>);
  profile(q, pop_back<decltype(q)>);
  profile(l, pop_back<decltype(l)>);
}
```

```
$ ./a.out
0 ms
0 ms
30 ms
```

2.6 Iterator

Table of Contents

- *Iterate an Internal Vector*
- *Iterate a file*
- *Position after Erasing*
- *Vector Comparision*

2.6.1 Upper Bound

Note that *std::upper_bound(x.begin(), x.end(), val)* finds an element which is *greater than the val*. However, *std::lower_bound(x.begin(), x.end(), val)*, finds an element which is *greater or equal to the val*.

```
#include <iostream>
#include <deque>
#include <algorithm>

int main(int argc, char *argv[]) {
  std::deque<int> v{1,2,3,4,5,7,10};

  auto x = 5;
  auto pos1 = std::upper_bound(v.begin(), v.end(), x);
  std::cout << *pos1 << "\n";

  auto pos2 = std::lower_bound(v.begin(), v.end(), x);
  std::cout << *pos2 << "\n";

  // 7
  // 5
}
```

2.6.2 Insert an Element into a Sorted List

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <range/v3/view/all.hpp>


int main(int argc, char *argv[]) {
  std::deque<int> v{1,2,3,4,5,7,10};

  auto x = 8;
  auto pos = std::upper_bound(v.begin(), v.end(), x);
  v.insert(pos, x);
  std::cout << ranges::views::all(v) << "\n";
  // [1,2,3,4,5,7,8,10]
}
```

2.6.3 Erase an Element in a Sorted List

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <range/v3/view/all.hpp>


int main(int argc, char *argv[]) {
  std::deque<int> v{1,2,3,4,5,7,10};

  auto x = 7;
  auto pos = std::lower_bound(v.begin(), v.end(), x);
  v.erase(pos);
  std::cout << ranges::views::all(v) << "\n";
  // [1,2,3,4,5,10]
}
```

2.6.4 Reverse Range-based for Loop

```
// via boost
// $ g++ --std=c++14 -Wall -Werror -g -O3 reverse.cpp
// $ ./a.out
// dlrow olleh

#include <iostream>
#include <string>
#include <boost/range/adaptor/reversed.hpp>

using namespace boost;

int main(int argc, char *argv[]) {
  std::string in = "hello world";
  std::string out;
  for (const auto &c : adaptors::reverse(in)) {
      out += c;
  }
  std::cout << out << "\n";
}
```

2.6.5 Customize an Iterator

```
// $ g++ -std=c++17 -Wall -Werror -g -O3 a.cc

#include <iostream>
#include <memory>

template <typename T>
class Array
{
```

(continues on next page)

(continued from previous page)

```
 public:
  class iterator
  {
    public:
      iterator(T *ptr) : ptr_{ptr} {}
      iterator operator++() { auto i = *this; ++ptr_; return i; }
      iterator operator++(int) { ++ptr_; return *this;};
      T &operator*() { return *ptr_; }
      T *operator->() { return ptr_; }
      bool operator==(const iterator &rhs) { return ptr_ == rhs.ptr_; }
      bool operator!=(const iterator &rhs) { return ptr_ != rhs.ptr_; }
    private:
      T *ptr_;
  };

  class const_iterator
  {
    public:
     const_iterator(T *ptr) : ptr_{ptr} {}
     const_iterator operator++() { auto i = *this; ++ptr_; return i; }
     const_iterator operator++(int) { ++ptr_; return *this; }
     const T &operator*() const { return *ptr_; }
     const T *operator->() const { return ptr_; }
     bool operator==(const const_iterator &rhs) { return ptr_ == rhs.ptr_; }
     bool operator!=(const const_iterator &rhs) { return ptr_ != rhs.ptr_; }
    private:
     T *ptr_;
  };

  Array(size_t size) : size_(size), data_{std::make_unique<T[]>(size)} {}
  size_t size() const { return size_; }
  T &operator[](size_t i) { return data_[i]; };
  const T &operator[](size_t i) const { return data_[i]; }
  iterator begin() { return iterator(data_.get()); }
  iterator end() { return iterator(data_.get() + size_); }
  const_iterator cbegin() const { return const_iterator(data_.get()); }
  const_iterator cend() const { return const_iterator(data_.get() + size_); }

 private:
  size_t size_;
  std::unique_ptr<T[]> data_;
};



int main(int argc, char *argv[])
{
  Array<double> points(2);
  points[0] = 55.66;
  points[1] = 95.27;
  for (auto &e : points) {
    std::cout << e << "\n";
```

(continues on next page)

(continued from previous page)

```
  }
  for (auto it = points.cbegin(); it != points.cend(); ++it) {
    std::cout << *it << "\n";
  }
}
```

2.6.6 Iterate an Internal Vector

```
#include <iostream>
#include <utility>
#include <vector>

template<typename T>
class Vector {
 public:
  using iterator = typename std::vector<T>::iterator;
  using const_iterator = typename std::vector<T>::const_iterator;

  inline iterator begin() noexcept {return v.begin();}
  inline iterator end() noexcept {return v.end();}
  inline const_iterator cbegin() const noexcept {return v.cbegin();}
  inline const_iterator cend() const noexcept {return v.cend();}

  template<class... Args>
  auto emplace_back(Args&&... args) {
      return v.emplace_back(std::forward<Args>(args)...);
  }
 private:
  std::vector<T> v;
};


int main(int argc, char *argv[]) {
  Vector<int> v;
  v.emplace_back(1);
  v.emplace_back(2);
  v.emplace_back(3);

  for (auto &it : v) {
      std::cout << it << std::endl;
  }
  return 0;
}
```

2.6.7 Iterate a file

```cpp
// $ g++ -std=c++17 -Wall -Werror -g -O3 a.cc
// $ ./a.out file

#include <iostream>
#include <iterator>
#include <fstream>
#include <string>

class line : public std::string {};

std::istream &operator>>(std::istream &is, line &l)
{
  std::getline(is, l);
  return is;
}

class FileReader
{
 public:
  using iterator = std::istream_iterator<line>;
  inline iterator begin() noexcept { return begin_; }
  inline iterator end() noexcept { return end_; }

 public:
  FileReader(const std::string path) : f_{path}, begin_{f_} {}
  friend std::istream &operator>>(std::istream &, std::string &);

 private:
  std::ifstream f_;
  iterator begin_;
  iterator end_;
};

int main(int argc, char *argv[])
{
  FileReader reader(argv[1]);
  for (auto &line : reader) {
    std::cout << line << "\n";
  }
}
```

2.6.8 Position after Erasing

```
// deque
    #include <iostream>
    #include <deque>
    #include <range/v3/view/all.hpp>

    int main(int argc, char *argv[]) {
      std::deque<int> q{1, 2, 3, 4, 5};
      auto it = q.begin() + 2;

      std::cout << *it << "\n";
      std::cout << ranges::views::all(q) << "\n";

      q.erase(it);
      std::cout << *it << "\n";
      std::cout << ranges::views::all(q) << "\n";

      // output
      //   3
      //   [1,2,3,4,5]
      //   4
      //   [1,2,4,5]
    }
```

```
#include <iostream>
#include <vector>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5};
  auto it = v.begin() + 2;

  std::cout << *it << "\n";
  std::cout << ranges::views::all(v) << "\n";

  v.erase(it);
  std::cout << *it << "\n";
  std::cout << ranges::views::all(v) << "\n";

  // output
  //   3
  //   [1,2,3,4,5]
  //   4
  //   [1,2,4,5]
}
```

```
#include <iostream>
#include <list>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
```

(continues on next page)

(continued from previous page)

```
  std::list<int> l{1, 2, 3, 4, 5};
  auto it = l.begin();
  ++it;

  std::cout << *it << "\n";
  std::cout << ranges::views::all(l) << "\n";

  // Note that Iterators, pointers and references referring to elements
  // removed by the function are invalidated. This is an example to show
  // that an iterator do not point to the next element after erasing.
  l.erase(it);
  std::cout << *it << "\n";
  std::cout << ranges::views::all(l) << "\n";
  // output
  //   2
  //   [1,2,3,4,5]
  //   2
  //   [1,3,4,5]
}
```

2.6.9 Vector Comparision

Note that the comparision operators are removed in C++20 (see [doc](https://en.cppreference.com/w/cpp/container/vector)). Therefore, using a lambda function as compare function is better than using default comparision when elements are not builtin types or has its own comparision operators.

```
#include <iostream>
#include <vector>

int main(int argc, char *argv[]) {
    std::vector<int> v1{5,2};
    std::vector<int> v2{2,3,4};
    std::cout << (v1 < v2) << "\n";
    // output: 0
}
```

2.7 Template

Table of Contents

- *Fold expressions*
- *Limit a Template Types*
- *Specialize Types*
- *Curiously recurring template pattern*
- *Parametric Expressions*
- *Template Template Parameters*
- *Access Protected Membors in Sub-Template*

2.7.1 Instantiate a Template

```
#include <iostream>

struct A {};
struct B {};

template <typename T, typename U>
struct Foo {
  Foo(T t, U u) : t_(t), u_(u) {}

  T t_;
  U u_;
};

template <typename F, typename T, typename U>
struct Bar {
  Bar(T t, U u) : f_(t, u) {}

  F f_;
};

// instantiate template Foo
template class Foo<A, B>;

int main() {
  Bar<Foo<A, B>, A, B>(A(), B());
  return 0;
}
```

2.7.2 Template Specialization

```
#include <iostream>

template <typename T, typename U>
class Base
{
private:
  T m_a;
  U m_b;

public:

  Base(T a, U b) : m_a(a), m_b(b) {};

  T foo() { return m_a; }
  U bar() { return m_b; }
};

// partial specialization
template<typename T>
class Base <T, int>
{
private:
  T m_a;
  int m_b;
public:
  Base(T a, int b) : m_a(a), m_b(b) {}
  T foo() { return m_a; }
  int bar() { return m_b; }
};

// full specialization
template<>
class Base <double, double>
{
private:
  double d_a;
  double d_b;
public:
  Base(double a, double b) : d_a(a), d_b(b) {}
  double foo() { return d_a; }
  double bar() { return d_b; }
};


int main (int argc, char *argv[])
{
  Base<float, int> foo(3.33, 1);
  Base<double, double> bar(55.66, 95.27);
  std::cout << foo.foo() << std::endl;
  std::cout << foo.bar() << std::endl;
  std::cout << bar.foo() << std::endl;
```

(continues on next page)

(continued from previous page)

```
  std::cout << bar.bar() << std::endl;
  return 0;
}
```

2.7.3 Class Template

```
#include <iostream>

template <typename T>
class Area
{
protected:
  T w;
  T h;
public:
  Area(T a, T b) : w(a), h(b) {}
  T get() { return w * h; }
};

class Rectangle : public Area<int>
{
public:
  Rectangle(int a, int b) : Area<int>(a, b) {}
};

template <typename T>
class GenericRectangle : public Area<T>
{
public:
  GenericRectangle(T a, T b) : Area<T>(a, b){}
};


int main (int argc, char *argv[])
{
  Rectangle r(2, 5);
  GenericRectangle<double> g1(2.5, 3.);
  GenericRectangle<int> g2(2, 3);

  std::cout << r.get() << std::endl;
  std::cout << g1.get() << std::endl;
  std::cout << g2.get() << std::endl;
  return 0;
}
```

2.7.4 Variadic Template (Parameter Pack)

```
#include <iostream>
#include <utility>
#include <vector>

template <typename T>
class Vector {
protected:
  std::vector<T> v;
public:

  template<typename ...Args>
  Vector(Args&&... args) {
    (v.emplace_back(std::forward<Args>(args)), ...);
  }

  using iterator = typename std::vector<T>::iterator;
  iterator begin() noexcept { return v.begin(); }
  iterator end() noexcept { return v.end(); }
};


int main(int argc, char *argv[]) {

  Vector<int> v{1,2,3};
  for (const auto &x : v)
  {
    std::cout << x << "\n";
  }
}
```

2.7.5 Fold expressions

```
// g++ -std=c++17 -Wall -Werror -O3 a.cc

#include <iostream>
#include <utility>

template <typename ...Args>
decltype(auto) f(Args&& ...args) {
  auto l = [](auto &&x) { return x * 2; };
  return (l(std::forward<Args>(args)) + ...);
}

int main(int argc, char *argv[]) {
  std::cout << f(1, 2, 3, 4, 5) << std::endl;
}
```

2.7.6 Limit a Template Types

```
#include <iostream>
#include <string>
#include <type_traits>

template<typename S,
  typename = typename std::enable_if<
    std::is_same<
      std::string,
      typename std::decay<S>::type
    >::value
  >::type
>
void Foo(S s) {
  std::cout << s << "\n";
}


int main(int argc, char *argv[]) {
  std::string s1 = "Foo";
  const std::string s2 = "Bar";
  Foo(s1);
  Foo(s2);

  // Foo(123);    compile error
  // Foo("Baz");  compile error
}
```

2.7.7 Specialize Types

```
#include <iostream>
#include <string>
#include <type_traits>

template<typename S>
void Foo(S s) {
  if (std::is_integral<S>::value) {
    std::cout << "do a task for integer..." << "\n";
    return;
  }
  if (std::is_same<std::string, typename std::decay<s>::type>::value)
  {
    std::cout << "do a task for string..." << "\n";
    return;
  }
}

int main(int argc, char *argv[]) {
  std::string s1 = "Foo";
  Foo(s1);
```

(continues on next page)

(continued from previous page)

```
  Foo(123);
}
```

Template Specialization approach

```
#include <iostream>
#include <string>
#include <type_traits>

template<typename S>
void Foo(S s) {}

template <>
void Foo<int>(int s) {
  std::cout << "do a task for integer..." << "\n";
}
template<>
void Foo<std::string>(std::string s) {
  std::cout << "do a task for string..." << "\n";
}


int main(int argc, char *argv[]) {
  std::string s1 = "Foo";
  Foo(s1);
  Foo(123);
}
```

2.7.8 Curiously recurring template pattern

```
#include <iostream>

// Curiously Recurring Template Pattern (CRTP)

template <typename D>
class Base
{
public:
  void interface() {
    static_cast<D *>(this)->implement();
  }

  static void static_interface() {
    D::static_interface();
  }

  void implement() {
    std::cout << "Base" << std::endl;
  }
};
```

(continues on next page)

(continued from previous page)

```
class DerivedFoo : public Base<DerivedFoo>
{
public:
  void implement() {
    std::cout << "Foo" << std::endl;
  }
  static void static_interface() {
    std::cout << "Static Foo" << std::endl;
  }
};

class DerivedBar : public Base<DerivedBar> {};

int main (int argc, char *argv[])
{
  DerivedFoo foo;
  DerivedBar bar;

  foo.interface();
  foo.static_interface();
  bar.interface();

  return 0;
}
```

2.7.9 Parametric Expressions

```
#include <iostream>

// g++ -std=c++17 -fconcepts -g -O3 a.cpp

decltype(auto) min(auto&& lhs, auto&& rhs) {
  return lhs < rhs ? lhs : rhs;
}

int main(int argc, char *argv[]) {
  std::cout << min(1, 2) << "\n";
  std::cout << min(3.14, 2.718) << "\n";
}
```

```
#include <iostream>

template<typename T>
decltype(auto) min(T&& lhs,T&& rhs) {
  return lhs < rhs ? lhs : rhs;
}

int main(int argc, char *argv[]) {
  std::cout << min(1, 2) << "\n";
```

(continues on next page)

(continued from previous page)

```
  std::cout << min(3.14, 2.718) << "\n";
}
```

```
#include <iostream>

auto min = [](auto&& lhs, auto&& rhs) {
  return lhs < rhs ? lhs : rhs;
};

int main(int argc, char *argv[]) {
  std::cout << min(1, 2) << "\n";
  std::cout << min(3.14, 2.718) << "\n";
}
```

Reference

_ Parametric Expressions

2.7.10 Template Template Parameters

```
#include <vector>
#include <deque>

template <template<class, class> class V, class T, class A>
void f(V<T, A> &v) {
  v.pop_back();
}

int main(int argc, char *argv[]) {
  std::vector<int> v{0};
  std::deque<int> q{1};
  f<std::vector, int>(v);
  f<std::deque, int>(q);
}
```

2.7.11 Access Protected Membors in Sub-Template

Accessing protected members by pulling the names into the current scope via `using`.

```
#include <iostream>

template <typename T>
class A {
 public:
  A(T p) : p_{p} {}
  decltype(auto) f() { std::cout << p_ << "\n"; }
 protected:
  T p_;
};
```

(continues on next page)

(continued from previous page)

```
template <typename T>
class B : A<T> {
  using A<T>::p_;
 public:
  B(T p) : A<T>(p) {}
  decltype(auto) g() { std::cout << p_ << "\n"; }
};

int main(int argc, char *argv[]) {
  A<int> a(0);
  B<int> b(0);
  a.f();
  b.g();
}
```

Another option is qualifying name via the `this` pointer.

```
#include <iostream>

template <typename T>
class A {
 public:
  A(T p) : p_{p} {}
  decltype(auto) f() { std::cout << p_ << "\n"; }
 protected:
  T p_;
};

template <typename T>
class B : A<T> {
 public:
  B(T p) : A<T>{p} {}
  decltype(auto) g() { std::cout << this->p_ << "\n"; }
};

int main(int argc, char *argv[]) {
  A<int> a(0);
  B<int> b(0);
  a.f();
  b.g();
}
```

2.8 Variadic

Table of Contents

2.8.1 Variadic Function

```cpp
#include <iostream>

template <typename T>
int sum(T x)
{
  return x;
}

template <typename T, typename ...Args>
int sum(T x, Args ...args)
{
  return x + sum(args...);
}

int main(int argc, char *argv[])
{
  std::cout << sum(1, 2, 3, 4, 5) << std::endl;
}
```

By using C++17 or above, Fold expression can simplify the previous snippet.

```cpp
#include <iostream>

template <typename ...Args>
int sum(Args ...args)
{
  return (args + ...);
}

int main(int argc, char *argv[])
{
  std::cout << sum(1, 2, 3, 4, 5) << std::endl;
}
```

2.8.2 Generic lambda expressions

C++14 allows lambda function using `auto` type-specifier in the arguments, which is similar to a template function.

```
#include <iostream>

template <typename T>
auto sum(T x)
{
  return x;
}

template <typename T, typename ...Args>
auto sum(T x, Args ...args)
{
  return x + sum(args...);
}

int main(int argc, char *argv[])
{
  auto s = [](auto ...args) { return sum(args...); };
  std::cout << s(1, 2, 3, 4, 5) << std::endl;
}
```

By using C++17 or above, a programmer can simplify the previous code as following snippet.

```
// g++ -std=c++17 -Wall -Werror -O3 a.cc

#include <iostream>

int main(int argc, char *argv[])
{
  auto sum = [](auto ...args) { return (args + ...); };
  std::cout << sum(1, 2, 3, 4, 5) << std::endl;
}
```

2.8.3 Variadic constructor

```
#include <iostream>
#include <vector>

class Foo {
 public:

  template <typename ...Args>
  Foo(Args ...args)
  {
    Sum(args...);
  }

  template <typename T>
  void Sum(T t)
```

(continues on next page)

(continued from previous page)

```
  {
    sum += t;
  }

  template <typename T, typename ...Args>
  void Sum(T t, Args ...args)
  {
    sum += t;
    Sum(args...);
  }

  void Print()
  {
    std::cout << sum << std::endl;
  }

 private:
   int sum = 0;
};

int main(int argc, char *argv[])
{
  auto f = Foo(1, 2, 3, 4, 5);
  f.Print();
}
```

```
#include <iostream>
#include <vector>

class Foo {
 public:

  template <typename T>
  Foo(T t)
  {
    sum += t;
  }

  template <typename T, typename ...Args>
  Foo(T t, Args ...args) : Foo(args...)
  {
    sum += t;
  }

  void Print()
  {
    std::cout << sum << std::endl;
  }

 private:
   int sum = 0;
};
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[])
{
  auto f = Foo(1, 2, 3, 4, 5);
  f.Print();
}
```

Warning: Please don't invoke a template constructor in a contructor because a new object will be created instead of updating the current object's status.

```
#include <iostream>
#include <vector>

class Foo {
 public:
  template <typename T>
  Foo(T t)
  {
    sum += t;
  }

  template <typename T, typename ...Args>
  Foo(T t, Args ...args)
  {
    sum += t;
    Foo(args...);
  }

  void Print()
  {
    std::cout << sum << std::endl;
  }

 private:
   int sum = 0;
};

int main(int argc, char *argv[])
{
  auto f = Foo(1, 2, 3, 4, 5);
  f.Print();
}
```

```
#include <iostream>
#include <vector>

class Foo {
 public:
  template <typename ...Args>
```

(continues on next page)

(continued from previous page)

```
  Foo(Args ...args)
  {
    sum = (args + ...);
  }

  void Print()
  {
    std::cout << sum << std::endl;
  }

 private:
   int sum = 0;
};

int main(int argc, char *argv[])
{
  auto f = Foo(1, 2, 3, 4, 5);
  f.Print();
}
```

2.8.4 Static Loop unrolling

```
#include <iostream>
#include <utility>

template <size_t N>
struct Loop {
  template <typename F, typename ...Args>
  static void run(F &&f, Args&& ...args)
  {
    Loop<N-1>::run(std::forward<F>(f),std::forward<Args>(args)...);
    f(args..., N-1);
  }
};

template <>
struct Loop<0> {
  template <typename F, typename ...Args>
  static void run(F &&f, Args&& ...args) {}
};

int main(int argc, char *argv[])
{
  size_t counter = 0;
  // for (int i = 0; i < 5; ++i) { counter += i; }
  Loop<5>::run([&](auto i) { counter += i; });
  std::cout << counter << std::endl;
}
```

2.8.5 Fold expression

```
#include <iostream>
#include <vector>

int main(int argc, char *argv[])
{
  [](auto ...args) {
    return (args + ...);
  }(1, 2, 3 ,4 ,5);

  std::vector<int> v;
  [](auto &&v, auto ...args) {
    (v.emplace_back(args), ...);
  }(v);

  [](auto ...args) {
    (std::cout << ... << args) << "\n";
  }(1, 2, 3, 4, 5);

  [](auto &&f, auto ...args) {
    return (... + f(args));
  }([](auto x) { return x * 2; }, 1, 2, 3, 4, 5);
}
```

2.9 Perfect Forwarding

Table of Contents

Perfect forwarding is a way for a programmer to design a generic wrapper function in their C++ programs. However, few examples show how to use it in a real scenario. Instead of explaining what a C++ *perfect forwarding* is, this article tries to collect use cases about using it.

2.9.1 Decorator Pattern

```
#include <iostream>
#include <utility>
#include <chrono>

template <typename Func, typename ...Args>
auto Decorator(Func &&f, Args&&... args) {

    const auto s = std::chrono::system_clock::now();
    auto ret = f(std::forward<Args>(args)...);
    const auto e = std::chrono::system_clock::now();
    std::chrono::duration<double> d = e - s;

    std::cout << "Time Cost: " << d.count() << std::endl;
    return ret;
}

long fib(long n) {
    return n < 2 ? 1 : fib(n-1) + fib(n-2);
}

int main() {
    Decorator(fib, 35);
    return 0;
}
```

2.9.2 Profiling

```
#include <iostream>
#include <utility>
#include <chrono>

class Timer {
public:
    Timer() : s_(std::chrono::system_clock::now()) {}
    ~Timer() {
        e_ = std::chrono::system_clock::now();
        std::chrono::duration<double> d = e_ - s_;
        std::cout << "Time Cost: " << d.count() << std::endl;
    }
private:
    std::chrono::time_point<std::chrono::system_clock> s_;
    std::chrono::time_point<std::chrono::system_clock> e_;
};

template <typename Func, typename ...Args>
auto Profile(Func f, Args&&... args) {
    Timer timer;
    return f(std::forward<Args>(args)...);
}
```

(continues on next page)

(continued from previous page)

```
long fib1(long n) {
    return (n < 2) ? 1 : fib1(n-1) + fib1(n-2);
}

template<long N>
struct f {
    static constexpr long v = f<N-1>::v + f<N-2>::v;
};

template<>
struct f<0> {
    static constexpr long v = 0;
};

template<>
struct f<1> {
    static constexpr long v = 1;
};

int main() {
    long ret = -1;
    ret = Profile(fib1, 35);
    std::cout << ret << std::endl;

    ret = Profile([](){ return f<35>::v; });
    std::cout << ret << std::endl;
    return 0;
}
```

2.9.3 Factory Pattern

```
#include <iostream>
#include <utility>
#include <string>
#include <memory>

struct PostgresqlConfig { /* implementation */ };
struct MysqlConfig { /* implementation */ };

template <typename DB>
class Session {
public:
    void connect(const std::string url) {
        static_cast<DB*>(this)->connect(url);
    }
};

class Postgresql : public Session<Postgresql> {
private:
```

(continues on next page)

(continued from previous page)

```
    PostgresqlConfig config_;
public:
    Postgresql(PostgresqlConfig c) : config_(c) {}

    void connect(const std::string url) {
        std::cout << "Connecting to Postgresql..." << std::endl;
        // connecting
    }
};

class Mysql : public Session<Mysql> {
private:
    MysqlConfig config_;
public:
    Mysql(MysqlConfig c) : config_(c) {}

    void connect(const std::string url) {
        std::cout << "Connecting to Mysql..." << std::endl;
        // connecting
    }
};

/**
 * An example of Perfect Forwarding
 */
template <typename S, typename C>
std::shared_ptr<S> SessionFactory(C&& c) {
    return std::make_shared<S>(std::forward<C>(c));
}

using PostgresSession = Session<Postgresql>;
using MysqlSession = Session<Mysql>;
using PostgresPtr = std::shared_ptr<PostgresSession>;
using MysqlPtr = std::shared_ptr<MysqlSession>;

int main(int argc, char *argv[]) {

    PostgresqlConfig pc;
    MysqlConfig mc;

    PostgresPtr ps = SessionFactory<Postgresql>(pc);
    MysqlPtr ms = SessionFactory<Mysql>(mc);

    ps->connect("postgresql://...");
    ms->connect("mysql://...");
    return 0;
}
```

2.10 Casting

Table of Contents

2.10.1 C Style Casting

```
#include <iostream>
#include <cmath>

int main(int argc, char *argv[]) {
  double x = M_PI;
  int xx = (int) x;
  std::cout << xx << "\n";

  long z = LONG_MAX;
  int zz = (int) z;  // dangerous. overflow
  std::cout << zz << "\n";
}
```

2.10.2 Const Casting

```
#include <iostream>

void f(const int &x) {
  const_cast<int &>(x) = 0;
}

int main(int argc, char *argv[]) {
  int x = 123;
  f(x);
  std::cout << x << "\n";
}
```

2.10.3 Reinterpret Casting

```
#include <iostream>
#include <iomanip>

struct A { int x; int y; };

int main(int argc, char *argv[]) {
  A a{1, 2};

  // convert a struct to a byte array
  char *buf = reinterpret_cast<char *>(&a);
  for (int i = 0; i < sizeof(A); ++i) {
    std::cout << static_cast<int>(buf[i]) << " ";
  }
  // output: 1 0 0 0 2 0 0 0
}
```

2.10.4 Static Casting

```
#include <iostream>
#include <memory>

struct A {
  virtual void f() { std::cout << __func__ << "\n"; }
  virtual ~A() = default;
};

struct B : public A {;
  B(int *x) : x_{x} {}
  int *g() { return x_; }
  int *x_;
};

int main(int argc, char *argv[]) {
  auto a = std::make_unique<A>();
  // downcasting may be dangerous
  auto b = static_cast<B *>(a.get());
  auto x = b->g();
  std::cout << *x << "\n";
}
```

2.10.5 Dynamic Casting

```cpp
#include <iostream>
#include <memory>

struct A {
  virtual void f() { std::cout << __func__ << "\n"; }
};

struct B : public A {
   void f() override { std::cout << __PRETTY_FUNCTION__ << "\n"; }
};

int main(int argc, char *argv[]) {
  auto a = std::make_unique<A>();
  auto b = std::make_unique<B>();

  // downcast
  auto bb = dynamic_cast<B *>(a.get());
  std::cout << "Is dynamic_cast(*a) to *b success? " << !!bb << "\n";
  // output: Is dynamic_cast(*a) to *b success? 0

  // upcast
  auto aa = dynamic_cast<A *>(b.get());
  std::cout << "Is dynamic_cast(*b) to *a success? " << !!aa << "\n";
  // output: Is dynamic_cast(*a) to *b success? 1
}
```

2.11 constexpr

Table of Contents

2.11.1 constexpr Function

```cpp
#include <iostream>
#include <utility>
#include <chrono>

class Timer {
public:
    inline void start() {
        start_ = std::chrono::system_clock::now();
    }
```

(continues on next page)

(continued from previous page)

```
    inline void end() {
        end_ = std::chrono::system_clock::now();
    }

    inline void out() {
        std::chrono::duration<double> d = end_ - start_;
        std::cout << "Time cost: " << d.count() << "\n";
    }
private:
    std::chrono::time_point<std::chrono::system_clock> start_;
    std::chrono::time_point<std::chrono::system_clock> end_;
};

constexpr long fib(long n) {
    return (n < 2) ? 1 : fib(n-1) + fib(n-2);
}

int main() {
    Timer timer;
    long n = 40;

    timer.start();
    int r1 = fib(n);
    timer.end();
    timer.out();

    timer.start();
    constexpr long r2 = fib(40);
    timer.end();
    timer.out();

    return 0;
}
```

output:

```
$ g++ -std=c++14 -g -O3 a.cpp
$ ./a.out
Time cost: 0.268229
Time cost: 8e-06
```

2.11.2 Compare to Metaprogramming

```
#include <iostream>
#include <utility>
#include "timer.h"

template <long N>
struct Fib {
    static long const v = Fib<N-1>::v + Fib<N-2>::v;
};

template <>
struct Fib<0> {
    static long const v = 1;
};

template <>
struct Fib<1> {
    static long const v = 1;
};

constexpr long fib(long n)
{
    return (n < 2) ? 1 : fib(n-1) + fib(n-2);
}

int main() {

    Timer timer;

    timer.start();
    constexpr long r1 = Fib<40>::v;
    timer.end();
    timer.out();

    timer.start();
    constexpr long r2 = fib(40);
    timer.end();
    timer.out();

    return 0;
}
```

output:

```
g++ -std=c++14 -g -O3 a.cpp
$ ./a.out
Time cost: 9.7e-06
Time cost: 9.2e-06
```

After C++14, constexpr functions can

- invoke other constexpr functions.
- have variables with a constant expression.

- include conditional expressions or loops.
- be implicit inline.
- not have static or thread_local data.

2.12 Lambda

Table of Contents

2.12.1 Callable Objects

```
#include <iostream>

class Fib {
public:
    long operator() (const long n) {
        return (n <= 2) ? 1 : operator()(n-1) + operator()(n-2);
    }
};

int main() {
    Fib fib;
    std::cout << fib(10) << "\n";
    return 0;
}
```

Lambda version

```
#include <iostream>
#include <functional>

int main() {
    std::function<long(long)> fib = [&](long n) {
        return (n <= 2) ? 1 : fib(n-1) + fib(n-2);
    };
    std::cout << fib(10) << "\n";
    return 0;
}
```

2.12.2 Default Arguments

```
#include <iostream>

int main(int argc, char *argv[]) {
    auto fib = [](long n=0) {
        long a = 0, b = 1;
        for (long i = 0; i < n; ++i) {
            long tmp = b;
            b = a + b;
            a = tmp;
        }
        return a;
    };
    std::cout << fib() << "\n";
    std::cout << fib(10) << "\n";
    return 0;
}
```

2.12.3 Captureless

```
#include <iostream>

int main() {
    long (*fib)(long) = [](long n) {
        long a = 0, b = 1;
        for (long i = 0; i < n; ++i) {
            long tmp = b;
            b = a + b;
            a = tmp;
        }
        return a;
    };
    std::cout << fib(10) << "\n";
    return 0;
}
```

2.12.4 Lambda capture initializers

```
// g++ -std=c++17 -Wall -Werror -O3 a.cc

#include <iostream>
#include <utility>
#include <memory>

int main(int argc, char *argv[])
{
  std::unique_ptr<int> p = std::make_unique<int>(5566);
  auto f = [x = std::move(p)]() { std::cout << *x << std::endl; };
  f();
}
```

2.12.5 Capture by `std::move`

```
#include <iostream>
#include <utility>

struct Foo {
    Foo() { std::cout << "Constructor" << "\n"; }
    ~Foo() { std::cout << "Destructor" << "\n"; }
    Foo(const Foo&) { std::cout << "Copy Constructor" << "\n"; }
    Foo(Foo &&) { std::cout << "Move Constructor" << "\n";}

    Foo& operator=(const Foo&) {
        std::cout << "Copy Assignment" << "\n";
        return *this;
    }
    Foo& operator=(Foo &&){
        std::cout << "Move Assignment" << "\n";
        return *this;
    }
};

int main(int argc, char *argv[]) {
    Foo foo;
    [f=std::move(foo)] { /* do some tasks here...*/ }();
}
```

2.12.6 Copy a Global into a Capture

```
#include <iostream>

int g = 1;

// copy a global to a capture
auto bar = [g=g]() { return g + 1; };
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
    int g = 10;
    std::cout << bar() << "\n";
}
```

2.12.7 constexpr by Default

```
#include <iostream>

int main() {
    auto fib = [](long n) {
        long a = 0, b = 1;
        for (long i = 0; i < n; ++i) {
            long tmp = b;
            b = a + b;
            a = tmp;
        }
        return a;
    };

    // constexpr by default is new in c++17
    static_assert(fib(10) == 55);
    return 0;
}
```

output:

```
$ g++ -std=c++17 -g -O3 a.cpp
```

2.12.8 Generic Lambda

```
#include <iostream>
#include <utility>

// g++ -std=c++17 -g -O3 a.cpp

class Sum {
public:
    template <typename ...Args>
    constexpr auto operator()(Args&& ...args) {
        // Fold expression (since c++17)
        return (std::forward<Args>(args) + ...);
    }
};

int main() {
    Sum sum;
    constexpr int ret = sum(1,2,3,4,5);
    std::cout << ret << std::endl;
```

(continues on next page)

(continued from previous page)

```
    return 0;
}
```

The snippet is equal to the following example

```
#include <iostream>
#include <utility>

int main() {
    auto sum = [](auto&& ...args) {
        return (std::forward<decltype(args)>(args) + ...);
    };
    constexpr int ret = sum(1,2,3,4,5);
    std::cout << ret << std::endl;
    return 0;
}
```

In c+20, lambda supports explicit template paramter list allowing a programmer to utilize parameters' type instead of using *decltype*.

```
#include <iostream>

// g++ -std=c++2a -g -O3 a.cpp

int main(int argc, char *argv[])
{
    auto sum = []<typename ...Args>(Args&&... args) {
        return (std::forward<Args>(args) + ...);
    };
    constexpr int ret = sum(1,2,3,4,5);
    std::cout << ret << std::endl;
    return 0;
}
```

2.12.9 Comparison Function

```
#include <iostream>
#include <string>
#include <map>

struct Cmp {
    template<typename T>
    bool operator() (const T &lhs, const T &rhs) const {
        return lhs < rhs;
    }
};

int main(int argc, char *argv[]) {

    // sort by keys
    std::map<int, std::string, Cmp> m;
```

(continues on next page)

(continued from previous page)

```
    m[3] = "Foo";
    m[2] = "Bar";
    m[1] = "Baz";

    for (auto it : m) {
        std::cout << it.first << ", " << it.second << "\n";
    }
    return 0;
}
```

```
#include <iostream>
#include <string>
#include <map>

bool cmp(const int &lhs, const int &rhs) {
    return lhs < rhs;
}

int main(int argc, char *argv[]) {

    // sort by keys
    std::map<int, std::string, decltype(&cmp)> m(cmp);

    m[3] = "Foo";
    m[2] = "Bar";
    m[1] = "Baz";

    for (auto it : m) {
        std::cout << it.first << ", " << it.second << "\n";
    }
    return 0;
}
```

```
#include <iostream>
#include <functional>
#include <string>
#include <map>

template<typename T>
using Cmp = std::function<bool(const T &, const T &)>;

template<typename T>
bool cmp(const T &lhs, const T &rhs) {
    return lhs < rhs;
}

int main(int argc, char *argv[]) {

    // sort by keys
    std::map<int, std::string, Cmp<int>> m(cmp<int>);
```

(continues on next page)

(continued from previous page)

```
    m[3] = "Foo";
    m[2] = "Bar";
    m[1] = "Baz";

    for (auto it : m) {
        std::cout << it.first << ", " << it.second << "\n";
    }
    return 0;
}
```

```
#include <iostream>
#include <string>
#include <map>

int main(int argc, char *argv[]) {

    auto cmp = [](auto &lhs, auto &rhs) {
        return lhs < rhs;
    };

    // sort by keys
    std::map<int, std::string, decltype(cmp)> m(cmp);

    m[3] = "Foo";
    m[2] = "Bar";
    m[1] = "Baz";

    for (auto it : m) {
        std::cout << it.first << ", " << it.second << "\n";
    }
    return 0;
}
```

2.12.10 Break Loops

```
#include <iostream>

int main(int argc, char *argv[]) {
    bool is_stoped = false;
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            std::cout << i + j << " ";
            if (i + j == 5) {
                is_stoped = true;
                break;
            }
        }
        if (is_stoped) {
            break;
```

(continues on next page)

(continued from previous page)

```
        }
    }
    std::cout << std::endl;
    return 0;
}
```

The previous example shows a common way to break multiple loops via a flag. However, the drawback is a programmer requires to maintain flags if code includes nested loops. By using a lambda function, it is convenient for developers to break nested loops through the return.

```
#include <iostream>

int main(int argc, char *argv[]) {
    [&] {
        for (int i = 0; i < 5; ++i) {
            for (int j = 0; j < 5; ++j) {
                std::cout << i + j << " ";
                if (i + j == 5) {
                    return;
                }
            }
        }
    }();
    std::cout << std::endl;
    return 0;
}
```

2.12.11 Callback

```
#include <iostream>

template<typename F>
long fib(long n, F f) {
    long a = 0, b = 1;
    for (long i = 0; i < n; ++i) {
        long tmp = b;
        b = a + b;
        a = tmp;
        f(a);
    }
    return a;
}

int main(int argc, char *argv[]) {
    fib(10, [](long res) {
        std::cout << res << " ";
    });
    std::cout << "\n";
    return 0;
}
```

```
#include <iostream>
#include <functional>

using fibcb = std::function<void(long x)>;

long fib(long n, fibcb f) {
    long a = 0, b = 1;
    for (long i = 0; i < n; ++i) {
        long tmp = b;
        b = a + b;
        a = tmp;
        f(a);
    }
    return a;
}

int main(int argc, char *argv[]) {
    fib(10, [](long res) {
        std::cout << res << " ";
    });
    std::cout << "\n";
    return 0;
}
```

Programmers can also use function pointers to define a functino's callback parameter. However, function pointers are only suitable for captureless lambda functions.

```
#include <iostream>
#include <functional>

using fibcb = void(*)(long n);

long fib(long n, fibcb f) {
    long a = 0, b = 1;
    for (long i = 0; i < n; ++i) {
        long tmp = b;
        b = a + b;
        a = tmp;
        f(a);
    }
    return a;
}

int main(int argc, char *argv[]) {
    fib(10, [](long res) {
        std::cout << res << " ";
    });
    std::cout << "\n";
    return 0;
}
```

2.12.12 Reference

1. Back to Basics: Lambdas from Scratch
2. Demystifying C++ lambdas

2.13 Time

Table of Contents

2.13.1 Timestamp

```
// g++ -std=c++17 -Wall -Werror -O3 a.cc

#include <iostream>
#include <chrono>

using milliseconds = std::chrono::milliseconds;
namespace chrono = std::chrono;

int main(int argc, char *argv[])
{
  auto now = std::chrono::system_clock::now();
  auto t = now.time_since_epoch();
  std::cout << chrono::duration_cast<milliseconds>(t).count() << "\n";
}
```

2.13.2 To chrono::chrono::time_point

```
#include <iostream>
#include <iomanip>
#include <chrono>
#include <ctime>

namespace chrono = std::chrono;
using ms = std::chrono::milliseconds;

int main(int argc, char *argv[])
{
  using namespace std::literals;
  auto s = 1602207217323ms;
  chrono::system_clock::time_point tp(s);
  std::time_t t = chrono::system_clock::to_time_t(tp);
  std::cout << std::put_time(std::gmtime(&t), "%FT%TZ") << "\n";
}
```

2.13.3 Duration

```
#include <iostream>
#include <chrono>

int main(int argc, char *argv[]) {
  using seconds = std::chrono::seconds;
  using namespace std::chrono_literals;
  auto now = std::chrono::system_clock::now();
  auto future = now + 1ms;
  auto t = std::chrono::system_clock::to_time_t(future);
  std::cout << "timestamp: " << std::ctime(&t);

  auto start = std::chrono::system_clock::now();
  auto end = std::chrono::system_clock::now() + 10s;
  auto duration = duration_cast<seconds>(end - start);
  std::cout << duration.count() << " sec\n";

  // output:
  // timestamp: Thu Oct  6 12:39:35 2022
  // 10 sec
}
```

2.13.4 Profiling

```
#include <iostream>
#include <chrono>

#include <unistd.h>

using milliseconds = std::chrono::milliseconds;
namespace chrono = std::chrono;

int main(int argc, char *argv[])
{
  auto start = std::chrono::steady_clock::now();
  sleep(3);
  auto end = std::chrono::steady_clock::now();
  auto d = end - start;
  std::cout << chrono::duration_cast<milliseconds>(d).count() << "\n";
}
```

2.13.5 Literals

```
#include <iostream>
#include <chrono>

using ms = std::chrono::milliseconds;
namespace chrono = std::chrono;

int main(int argc, char *argv[])
{
  using namespace std::literals;
  auto t = 1602207217323ms;
  std::cout << std::chrono::duration_cast<ms>(t).count() << "\n";
}
```

2.13.6 Format Time

```
#include <iostream>
#include <iomanip>
#include <ctime>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  std::time_t t = std::time(nullptr);
  constexpr char fmt[] = "%c %Z";
  std::cout << "UTC " << std::put_time(std::gmtime(&t), fmt) << "\n";
  std::cout << "Local " << std::put_time(std::localtime(&t), fmt) << "\n";

  std::string tz = "America/Chicago";
  putenv(tz.data());
```

(continues on next page)

(continued from previous page)

```
  std::cout << "Chicago " << std::put_time(std::localtime(&t), fmt) << "\n";
}
```

2.13.7 To time_t

```
#include <iostream>
#include <iomanip>
#include <chrono>
#include <ctime>

namespace chrono = std::chrono;

int main(int argc, char *argv[])
{
  auto now = chrono::system_clock::now();
  std::time_t t = std::chrono::system_clock::to_time_t(now);
  std::cout << std::put_time(std::gmtime(&t), "%FT%TZ") << "\n";
}
```

2.13.8 ISO 8601 format

```
#include <iostream>
#include <iomanip>
#include <chrono>
#include <ctime>

namespace chrono = std::chrono;

int main(int argc, char *argv[])
{
  auto now = chrono::system_clock::now();
  std::time_t t = std::chrono::system_clock::to_time_t(now);
  std::cout << std::put_time(std::gmtime(&t), "%Y-%m-%dT%H:%M:%SZ") << "\n";
  std::cout << std::put_time(std::gmtime(&t), "%FT%TZ") << "\n";
  std::cout << std::put_time(std::gmtime(&t), "%FT%TZ%z") << "\n";
}
```

2.14 Smart Pointers

Table of Contents

2.14.1 Custom Deleters

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <iostream>
#include <string>
#include <exception>
#include <memory>

using FilePtr = std::unique_ptr<FILE, int (*)(FILE *)>;

constexpr void assert_that(bool statement, const char *msg) {
    if (!statement) {
        throw std::runtime_error(msg);
    }
}

int main(int argc, char *argv[]) {

    assert_that(argc == 2, "Usage: command [path]");

    FILE *f = nullptr;
    f = fopen(argv[1], "r+");
    assert_that(f, strerror(errno));

    // assign FILE* to a unique_ptr
    FilePtr fptr{f, fclose};
    assert_that(!!fptr, strerror(errno));
    assert_that(fseek(fptr.get(), 0, SEEK_END) == 0, strerror(errno));

    long size = ftell(fptr.get());
    assert_that(size >=0, strerror(errno));
    rewind(fptr.get());

    // using unique_ptr to create a buffer instead of using malloc
    std::unique_ptr<char[]> buf{ new char[size + 1]{0} };
    assert_that(!!buf, strerror(errno));

    size_t r = fread(buf.get(), 1, size, fptr.get());
    assert_that(r == size, "Reading error");
    std::cout << buf.get();
end:
    return 0;
}
```

2.14.2 `std::make_shared` and `std::make_unique`

`std::make_shared` and `std::make_unique` are the recommended ways to create smart pointers because compilers do guarantee the order of executions, which may introduce memory leaks when an exception is thrown. For example, the compilers may call `new T`, then `raise()`, and so on before `foo` is called. In this case, `std::unique_ptr` does not know the pointer `T` yet, so it is still on the heap.

```
using uptr = std::unique_ptr<T>;

bool raise() {
    throw std::exception();
    return true;
}

foo(uptr(new T), raise(), uptr(new T));
```

2.15 Return Value Optimization (RVO)

Table of Contents

2.15.1 Before starting

```
// foo.h

#include <iostream>

struct Foo {
    Foo() {
        std::cout << "Constructor" << "\n";
    }
    ~Foo() {
        std::cout << "Destructor" << "\n";
    }
    Foo(const Foo&) {
        std::cout << "Copy Constructor" << "\n";
    }
    Foo(Foo &&) {
        std::cout << "Move Constructor" << "\n";
    }
    Foo& operator=(const Foo&) {
        std::cout << "Copy Assignment" << "\n";
        return *this;
    }
    Foo& operator=(Foo &&){
        std::cout << "Move Assignment" << "\n";
        return *this;
    }
};
```

2.15.2 Return Value Optimization

```
#include "foo.h"

Foo FooRVO() {
    return Foo();
}

int main(int argc, char *argv[]) {
    Foo f = FooRVO();
}
```

2.15.3 Named Return Value Optimization

```
#include "foo.h"

Foo FooNRVO() {
    Foo foo;
    return foo;
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
    Foo f = FooNRVO();
}
```

2.15.4 Copy Elision

```
#include "foo.h"

void CopyElision(Foo foo) {}

int main(int argc, char *argv[]) {
    CopyElision(Foo());
}
```

2.15.5 Return a Global (w/o RVO)

```
#include "foo.h"

const Foo foo;

Foo ReturnGlobal() {
    return foo;
}

int main(int argc, char *argv[]) {
    Foo f = ReturnGlobal();
}
```

2.15.6 Return a Parameter (w/o RVO)

```
#include "foo.h"

Foo ReturnParam(Foo foo) {
    return foo;
}

int main(int argc, char *argv[]) {
    Foo f = ReturnParam(Foo());
}
```

2.15.7 Runtime Decision (w/ RVO)

```
#include "foo.h"

Foo FooRVO(bool is_x) {
    return is_x ? Foo() : Foo();
}

int main(int argc, char *argv[]) {
    Foo foo = FooRVO(true);
}
```

2.15.8 Runtime Decision (w/ RVO, w/o NRVO)

```
#include "foo.h"

Foo RVOButNoNRVO(bool is_x) {
    Foo x;
    return is_x ? x : Foo();
}

int main(int argc, char *argv[]) {
    Foo f = RVOButNoNRVO(false);
}
```

2.15.9 Runtime Decision (w/o NRVO)

```
#include "foo.h"

Foo FooNoNRVO(bool is_x) {
    Foo x, y;
    return is_x ? x : y;
}

int main(int argc, char *argv[]) {
    Foo foo = FooNoNRVO(true);
}
```

2.15.10 Return by `std::move` (w/o RVO)

```
#include "foo.h"

#include <utility>

Foo FooMove() {
    return std::move(Foo());
}
```

(continues on next page)

(continued from previous page)

```
int main(int argc, char *argv[]) {
    Foo foo = FooMove();
}
```

2.15.11 Return by `std::move` (w/o NRVO)

```
#include "foo.h"

#include <utility>

Foo FooMove() {
    Foo foo;
    return std::move(foo);
}

int main(int argc, char *argv[]) {
    Foo foo = FooMove();
}
```

2.15.12 Return a Member (w/o RVO)

```
#include "foo.h"

struct Bar {
    Foo foo;
};

Foo ReturnMember() {
    return Bar().foo;
}

int main(int argc, char *argv[]) {
    Foo f = ReturnMember();
}
```

2.16 Data Structure & Algorithm

Table of Contents

- *std::foreach*
- *std::find*
- *std::find_if & std::find_if_not*
- *std::transform*
- *std::generate*

2.16.1 Remove elements by conditions

```
#include <vector>

int main(int argc, char *argv[])
{
  std::vector<int> v{1, 2, 3, 4, 5, 6};
  for (auto it = v.begin(); it != v.end();) {
    if (*it > 3) {
      it = v.erase(it);
    } else {
      ++it;
    }
  }
}
```

2.16.2 Remove keys by conditions

```
#include <string>
#include <map>

int main(int argc, char *argv[])
{
  std::map<int, std::string> m{{1, "1"}, {2, "2"}, {3, "3"}};
  for (auto it = m.begin(); it != m.end();) {
    if (it->first > 1) {
      it = m.erase(it);
    } else {
      ++it;
    }
  }
}
```

2.16.3 std::map sort by key

```cpp
// g++ -std=c++17 -Wall -Werror -O3 a.cc

#include <iostream>
#include <map>

int main(int argc, char *argv[])
{
  // ascending
  std::map<int, int, std::less<int>> a{{3, 3}, {2, 2}, {1, 1}};
  // descending
  std::map<int, int, std::greater<int>> d{{3, 3}, {2, 2}, {1, 1}};

  auto print = [](auto &m) {
    for (const auto &[k, v] : m) {
      std::cout << k << " " << v << "\n";
    }
  };
  print(a); // 1, 2, 3
  print(d); // 3, 2, 1
}
```

2.16.4 std::map with object as key

```cpp
#include <iostream>
#include <map>

struct Data {
  int a;
  int b;
};

int main(int argc, char *argv[])
{
  auto cmp = [](auto &x, auto &y) { return x.a < y.b; };
  std::map<Data, int, decltype(cmp)> m{cmp};
  m[Data{1, 2}] = 1;
  m[Data{3, 4}] = 4;

  for (const auto &[k, v] : m) {
    std::cout << k.a << " " << k.b << " " << v << "\n";
  }
}
```

2.16.5 std::foreach

```
#include <iostream>
#include <vector>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::vector v{1, 2, 3};
  std::for_each(v.begin(), v.end(), [](auto &i) { i = i * 2; });
  std::for_each(v.begin(), v.end(), [](auto &i) { std::cout << i << "\n"; });
}
```

2.16.6 std::find

`std::find` returns an iterator to the first element in an array like object.

```
#include <iostream>
#include <vector>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::vector v{1, 2, 3};

  // complexity O(n)
  auto it = std::find(v.begin(), v.end(), 2);
  std::cout << *it << "\n";
}
```

2.16.7 std::find_if & std::find_if_not

```
#include <iostream>
#include <vector>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::vector v{1, 2, 3};
  auto x = find_if(v.begin(), v.end(), [](auto &i) { return i == 2; });
  std::cout << *x << "\n";

  auto y = find_if_not(v.begin(), v.end(), [](auto &i) { return i == 2; });
  std::cout << *y << "\n";
}
```

2.16.8 std::transform

```
#include <iostream>
#include <vector>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::string s = "Hello World";

  // transform elements in place
  std::transform(s.begin(), s.end(), s.begin(), ::toupper);
  std::cout << s << "\n";

  // transform elements and store in another object
  std::string o(s.size(), 0);
  std::transform(s.begin(), s.end(), o.begin(), ::tolower);
  std::cout << o << "\n";
}
```

2.16.9 std::generate

```
#include <iostream>
#include <random>
#include <vector>
#include <algorithm>

int main(int argc, char *argv[])
{
  std::random_device dev;
  std::mt19937 rng(dev());
  std::uniform_int_distribution<std::mt19937::result_type> dist(1,10);

  // generate a sequence
  std::vector<int> v(5);
  std::generate(v.begin(), v.end(), [&] { return dist(rng); });
  for (const auto &i : v) {
    std::cout << i << std::endl;
  }
}
```

2.17 coroutine

Table of Contents

– *Boost ASIO Echo Server*

2.17.1 Generator

```
// g++ -O3 -std=c++20 -Wall -Werror co.cc

#include <iostream>
#include <coroutine>

template <typename T>
class generator {
public:
  struct promise_type;
  using handle_type = std::coroutine_handle<promise_type>;
  handle_type h_;

  struct promise_type {
    T value_;
    std::exception_ptr exception_;
    generator<T> get_return_object() {
      return { handle_type::from_promise(*this) };
    }
    void unhandled_exception() { exception_ = std::current_exception(); }
    void return_void() {}
    std::suspend_always initial_suspend() noexcept { return {}; }
    std::suspend_always final_suspend() noexcept { return {}; }
    std::suspend_always yield_value(T v) noexcept {
      value_ = std::move(v);
      return {};
    }
  };

public:

  generator(handle_type h) : h_(h) {}
  ~generator() { h_.destroy(); }
  explicit operator bool() {
    next();
    return !h_.done();
  }

  T operator() () {
    next();
    cached_ = false;
    return std::move(h_.promise().value_);
  }

private:
  bool cached_ = false;
  void next() {
    if (cached_) {
```

(continues on next page)

(continued from previous page)

```
      return;
    }
    h_();
    if (h_.promise().exception_) {
      std::rethrow_exception(h_.promise().exception_);
    }
    cached_ = true;
  }
};

generator<uint64_t> fib(uint64_t n) {
  uint64_t a = 0, b = 1;
  for (uint64_t i = 0; i <= n; ++i) {
    co_yield a;
    uint64_t t = b;
    b = a + b;
    a = t;
  }
}

int main(int argc, char *argv[]) {
  auto g = fib(10);
  while (g) {
    std::cout << g() << " ";
  }
  // ./a.out
  //  0 1 1 2 3 5 8 13 21 34 55
}
```

2.17.2 Boost ASIO Echo Server

```
#include <iostream>
#include <boost/asio/co_spawn.hpp>
#include <boost/asio/detached.hpp>
#include <boost/asio/io_context.hpp>
#include <boost/asio/ip/tcp.hpp>
#include <boost/asio/signal_set.hpp>
#include <boost/asio/write.hpp>

using boost::asio::ip::tcp;
using boost::asio::awaitable;
using boost::asio::co_spawn;
using boost::asio::detached;
using boost::asio::use_awaitable;
namespace this_coro = boost::asio::this_coro;

constexpr uint64_t BUFSIZE = 1024;

awaitable<void> echo(tcp::socket &socket) {
  for (;;) {
```

(continues on next page)

(continued from previous page)

```
    char data[BUFSIZE] = {0};
    auto n = co_await socket.async_read_some(boost::asio::buffer(data), use_awaitable);
    co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
  }
}

awaitable<void> handle(tcp::socket socket) {
  try {
    co_await echo(socket);
  } catch(const std::exception &e) {
    std::cerr << e.what();
  }
}

awaitable<void> listener() {
  auto e = co_await this_coro::executor;
  tcp::acceptor acceptor(e, {tcp::v4(), 8888});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(e, handle(std::move(socket)), detached);
  }
}

int main(int argc, char *argv[]) {
  boost::asio::io_context io_context;
  boost::asio::signal_set signals(io_context, SIGINT, SIGTERM);
  signals.async_wait([&](auto, auto){ io_context.stop(); });
  co_spawn(io_context, listener(), detached);
  io_context.run();
}
```

```
# CMakeLists.txt
cmake_minimum_required(VERSION 3.10)
set(target a.out)
set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)
find_package(Boost)
add_executable(${target} a.cc)
target_include_directories(${target} PRIVATE "${CMAKE_CURRENT_SOURCE_DIR}")
target_include_directories(${target} PRIVATE "${Boost_INCLUDE_DIR}")
target_link_libraries(${target} ${Boost_LIBRARIES})
target_link_libraries(${target} INTERFACE Boost::coroutine)
```

2.18 Ranges

Table of Contents

- *c++20 range - filter*
- *c++20 range - split*
- *c++20 range - join*

2.18.1 range-v3 - debug a vector

```cpp
// g++ -O3 -std=c++17 -Wall -Werror -I${HDR} a.cpp

#include <iostream>
#include <vector>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{5, 4, 3, 2, 1, 1, 1};
  std::cout << ranges::views::all(v) << "\n";
  // [5,4,3,2,1,1,1]
}
```

2.18.2 range-v3 - all_of

```cpp
#include <iostream>
#include <vector>
#include <range/v3/algorithm/all_of.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{0, 2, 4};
  std::cout << ranges::all_of(v, [](auto &&x) { return !(x % 2); });
  // 1
}
```

```
>>> a = [0, 2, 4]
>>> all(x % 2 == 0 for x in a)
True
```

2.18.3 range-v3 - any_of

```cpp
#include <iostream>
#include <vector>
#include <range/v3/algorithm/any_of.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{0, 1, 2};
  std::cout << ranges::any_of(v, [](auto &&x) { return !(x % 2); });
  // 1
}
```

```
>>> a = [0, 1 ,2]
>>> any(x % 2 == 0 for x in a)
True
```

2.18.4 range-v3 - slice

```
#include <iostream>
#include <vector>
#include <range/v3/algorithm/copy.hpp>
#include <range/v3/action/slice.hpp>
#include <range/v3/view/slice.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> x{5, 4, 3, 2, 1};

  auto y = x | ranges::copy | ranges::actions::slice(1, 3);
  std::cout << ranges::views::all(y) << "\n";
  // [4,3]

  for (auto &&e : x | ranges::views::slice(2, 4)) {
    std::cout << e << "\n";
  }
}
```

```
>>> a = [5,4,3,2,1]
>>> print(a[1:3])
[4, 3]
```

2.18.5 range-v3 - enumerate

```
#include <iostream>
#include <vector>
#include <range/v3/view/enumerate.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{5, 4, 3, 2, 1, 1, 1};
  for (auto &&[i, e] : v |  ranges::views::enumerate) {
    std::cout << i << ", " << e << "\n";
  }
}
```

```
>>> a = [5,4,3,2,1,1]
>>> for i, e in enumerate(a):
...     print(i, e)
...
0 5
1 4
2 3
```

(continues on next page)

(continued from previous page)

```
3 2
4 1
5 1
```

2.18.6 range-v3 - concat vectors

```cpp
#include <iostream>
#include <vector>
#include <range/v3/view/concat.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> x{1, 5};
  std::vector<int> y{2, 8};
  std::vector<int> z{0, 3};
  auto r = ranges::views::concat(x, y, z);
  std::cout << ranges::views::all(r) << "\n";
  // [1,5,2,8,0,3]
}
```

```
>>> a = [1, 5]
>>> b = [2, 8]
>>> c = [0, 3]
>>> print(a + b + c)
[1, 5, 2, 8, 0, 3]
```

2.18.7 range-v3 - accumulate (sum)

```cpp
#include <iostream>
#include <vector>
#include <range/v3/numeric/accumulate.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5};
  const auto r = ranges::accumulate(v, 0);
  std::cout << r << "\n";
  // 15
}
```

```
>>> a = [1, 2, 3, 4, 5]
>>> sum(a)
15
```

2.18.8 range-v3 - accumulate (reduce)

```
#include <iostream>
#include <vector>
#include <range/v3/numeric/accumulate.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5};
  const auto r = ranges::accumulate(v, 1, [](auto &a, auto &b){
    return a + b;
  });
  std::cout << r << "\n";
  // 120
}
```

```
>>> from functools import reduce
>>> reduce(lambda x, y: x * y, [1, 2, 3, 4, 5], 1)
120
```

2.18.9 range-v3 - sort

```
#include <iostream>
#include <vector>
#include <range/v3/action/sort.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{5, 4, 3, 2, 1, 1, 1};
  v |= ranges::actions::sort;
  std::cout << ranges::views::all(v) << "\n";
  // [1,1,1,2,3,4,5]
}
```

```
>>> a = [5,4,3,2,1,1,1]
>>> a.sort()
>>> a
[1, 1, 1, 2, 3, 4, 5]
```

2.18.10 range-v3 - reverse sort

```
#include <iostream>
#include <vector>
#include <range/v3/action/sort.hpp>
#include <range/v3/action/reverse.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 5, 3, 2, 6};
```

(continues on next page)

(continued from previous page)

```
  v |= ranges::actions::sort | ranges::actions::reverse;
  std::cout << ranges::views::all(v) << "\n";
}
```

```
>>> a = [1, 5, 3, 2, 6]
>>> a.sort(reverse=True)
>>> a
[6, 5, 3, 2, 1]
```

2.18.11 range-v3 - sort & uniqe

```
// echo 5 4 3 2 1 1 1 | tr -s " " "\n" | sort | uniq

#include <iostream>
#include <vector>
#include <range/v3/action/unique.hpp>
#include <range/v3/action/sort.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{5, 4, 3, 2, 1, 1, 1};
  v |= ranges::actions::sort | ranges::actions::unique;
  std::cout << ranges::views::all(v) << "\n";
  // [1,2,3,4,5]
}
```

```
>>> a = [5, 4, 3, 2, 1, 1, 1]
>>> a = list({x for x in a})
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

2.18.12 range-v3 - zip

```
#include <iostream>
#include <vector>
#include <range/v3/view/zip.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> x{5, 4, 3, 2};
  std::vector<int> y{1, 2, 3 ,4};

  for (auto &&[a, b] : ranges::views::zip(x, y)) {
    std::cout << a << " " << b << "\n";
  }
}
```

```
>>> a = [5,4,3,2]
>>> b = [1,2,3,4]
>>> for x, y in zip(a, b):
...     print(x, y)
...
5 1
4 2
3 3
2 4
```

2.18.13 range-v3 - split

```
#include <iostream>
#include <vector>
#include <string>
#include <range/v3/view/c_str.hpp>
#include <range/v3/action/split.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::string s = "hello c++";
  auto v = ranges::actions::split(s, ranges::views::c_str(" "));
  std::cout << ranges::views::all(v) << "\n";
  // [hello,c++]
}
```

```
>>> s = "hello python"
>>> s.split(" ")
['hello', 'python']
```

2.18.14 range-v3 - tokenize

```
#include <iostream>
#include <vector>
#include <string>
#include <regex>
#include <range/v3/view/tokenize.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  const std::string s = "hello cpp";
  const auto p = std::regex{"[\\w]+"};
  auto r = s | ranges::views::tokenize(p);
  std::cout << ranges::views::all(r) << "\n";
}
```

```
>>> import re
>>> s = "hello python"
```

(continues on next page)

(continued from previous page)

```
>>> [m.group() for m in re.finditer(r"\w+", s)]
['hello', 'python']
```

2.18.15 range-v3 - join

```
#include <iostream>
#include <vector>
#include <string>
#include <range/v3/core.hpp>
#include <range/v3/view/join.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<std::string> v{"hello", "c++"};
  auto s = v | ranges::views::join(' ') | ranges::to<std::string>();
  std::cout << s << "\n";
}
```

```
>>> v = ['hello', 'python']
>>> ' '.join(v)
'hello python'
```

2.18.16 range-v3 - iota

```
#include <iostream>
#include <range/v3/view/iota.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  auto seq = ranges::views::iota(5, 8);
  std::cout << ranges::views::all(seq) << "\n";
  // [5,6,7]
}
```

```
>>> [x for x in range(5, 8)]
[5, 6, 7]
```

2.18.17 range-v3 - generate

```
#include <iostream>
#include <vector>
#include <range/v3/view/generate.hpp>
#include <range/v3/view/take.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
```

(continues on next page)

(continued from previous page)

```
  auto fib = ranges::views::generate([i=0, j=1]() mutable {
    int tmp = i; i+= j; j = i; return tmp;
  });

  auto v = fib | ranges::views::take(5);
  std::cout << ranges::views::all(v) << std::endl;
  // [0,1,2,4,8]
}
```

```
>>> def fib(n):
...     a, b = 0, 1
...     for _ in range(n):
...         yield a
...         a, b = b, a + b
...
>>> [x for x in fib(5)]
[0, 1, 1, 2, 3]
```

2.18.18 range-v3 - take

```
#include <iostream>
#include <range/v3/view/iota.hpp>
#include <range/v3/view/take.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  auto v = ranges::views::iota(5, 10) | ranges::views::take(3);
  std::cout << ranges::views::all(v) << "\n";
  // [5,6,7]
}
```

2.18.19 range-v3 - take_while

```
#include <iostream>
#include <range/v3/view/iota.hpp>
#include <range/v3/view/take_while.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  auto v = ranges::views::iota(5, 10)
      | ranges::views::take_while([](auto &&x) { return x < 8; });
  std::cout << ranges::views::all(v) << "\n";
}
```

2.18.20 range-v3 - drop

```
#include <iostream>
#include <vector>
#include <range/v3/action/drop.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5, 6};
  v |= ranges::actions::drop(3);
  std::cout << ranges::views::all(v) << "\n";
}
```

2.18.21 range-v3 - drop_while

```
#include <iostream>
#include <range/v3/view/iota.hpp>
#include <range/v3/view/drop_while.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  auto v = ranges::views::iota(5, 10)
      | ranges::views::drop_while([](auto &&x) { return x < 8; });
  std::cout << ranges::views::all(v) << "\n";
}
```

2.18.22 range-v3 - transform (map)

```
#include <iostream>
#include <vector>
#include <range/v3/view/transform.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5};
  auto r = v | ranges::views::transform([](auto &&x){ return x*x; });
  std::cout << ranges::views::all(r) << "\n";
  // [1,4,9,16,25]
}
```

2.18.23 range-v3 - filter

```
#include <iostream>
#include <vector>
#include <range/v3/view/filter.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3, 4, 5};
  auto r = v | ranges::views::filter([](auto &&x){ return x > 3; });
  std::cout << ranges::views::all(r) << "\n";
  // [4,5]
}
```

2.18.24 range-v3 - group_by

```
#include <iostream>
#include <string>
#include <range/v3/view/group_by.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::string s = "aaaabbbccd";
  auto r = s | ranges::views::group_by([](auto &&x, auto &&y){
    return x == y;
  });
  std::cout << ranges::views::all(r) << "\n";
  // [[a,a,a,a],[b,b,b],[c,c],[d]]
}
```

2.18.25 range-v3 - cycle

```
#include <iostream>
#include <vector>
#include <range/v3/view/cycle.hpp>
#include <range/v3/view/take.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3};
  auto r = v | ranges::views::cycle | ranges::views::take(6);
  std::cout << ranges::views::all(r) << "\n";
}
```

2.18.26 range-v3 - keys

```
#include <iostream>
#include <unordered_map>
#include <range/v3/view/map.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::unordered_map<int, int> m{{9, 5}, {2, 7}};
  auto keys = m | ranges::views::keys;
  for (auto &&k : keys) {
    std::cout << k << "\n";
  }
}
```

2.18.27 range-v3 - values

```
#include <iostream>
#include <unordered_map>
#include <range/v3/view/map.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::unordered_map<int, int> m{{9, 5}, {2, 7}};
  auto values = m | ranges::views::values;
  for (auto &&v : values) {
    std::cout << v << "\n";
  }
}
```

2.18.28 range-v3 - cartesian_product

```
#include <iostream>
#include <vector>
#include <string>
#include <range/v3/view/cartesian_product.hpp>

int main(int argc, char *argv[]) {
  std::string x = "ab";
  std::vector<int> y{1, 2};
  auto r = ranges::views::cartesian_product(x, y);
  for (auto &&[a, b] : r) {
    std::cout << a << b << "\n";
  }
  // a1 a2 b1 b2
}
```

2.18.29 range-v3 - permutation

```
#include <iostream>
#include <vector>
#include <range/v3/algorithm/permutation.hpp>
#include <range/v3/view/all.hpp>

int main(int argc, char *argv[]) {
  std::vector<int> v{1, 2, 3};
  do {
    std::cout << ranges::views::all(v) << "\n";
  } while (ranges::next_permutation(v));
}
```

2.18.30 c++20 range - iota

```
// g++-10 -Wall -Werror -O3 -g --std=c++20 a.cc

#include <iostream>
#include <ranges>

int main(int argc, char *argv[])
{
  using namespace std::ranges;

  for (auto i : views::iota(1) | views::take(5)) {
    std::cout << i << std::endl;
  }
}
```

2.18.31 c++20 range - transform

```
#include <iostream>
#include <ranges>
#include <vector>

int main(int argc, char *argv[])
{
  using namespace std::ranges;

  std::vector v{1, 2, 3};
  auto adaptor = views::transform([](auto &e) { return e * e; });
  for (auto i : v | adaptor) {
    std::cout << i << std::endl;
  }
}
```

2.18.32 c++20 range - filter

```
#include <iostream>
#include <ranges>
#include <vector>

int main(int argc, char *argv[])
{
  using namespace std::ranges;

  std::vector v{1, 2, 3};
  auto adaptor = views::filter([](auto &e) { return e % 2 == 0; });

  for (auto i : v | adaptor) {
    std::cout << i << std::endl;
  }
}
```

2.18.33 c++20 range - split

```
#include <iostream>
#include <ranges>
#include <string>

int main(int argc, char *argv[])
{
  using namespace std::ranges;
  std::string s{"This is a string."};

  for (auto v : s | views::split(' ')) {
    std::string w;
    for (auto &c : v) {
      w += c;
    }
    std::cout << w << std::endl;
  }
}
```

2.18.34 c++20 range - join

```
#include <iostream>
#include <ranges>
#include <vector>
#include <string>

int main(int argc, char *argv[])
{
  using namespace std::ranges;
  std::vector<std::string> v{"This", " ", "is", " ", "a", " ", "string."};
  std::string s;
```

(continues on next page)

(continued from previous page)

```
  for (auto &c : v | views::join) {
    s += c;
  }
  std::cout << s << std::endl;
}
```

CHAPTER

THREE

BASH CHEAT SHEET

3.1 Bash Basic cheatsheet

Table of Contents

3.1.1 :

true was instead simply aliased to :, and false like *let 0*.

```
$ while :; do sleep 1; date; done
```

3.1.2 Special Parameters

```
#!/bin/bash

foo() {
  # expand to the position params, equal to "$1$2..."
  echo $*
  # expand to the position params, equal to "$1" "$2" ...
  echo $@
  # expand to the number of position params
  echo $#
  # expand to the pid
  echo $$
  # expand to the exit status
  echo $?
  # expand to the name of shell script
  echo $0
}

foo "a" "b" "c"
```

3.1.3 Set Positional Parameters

```
$ set -- a b c
$ echo $@
a b c
$ echo "$1" "$2" "$3"
a b c
```

3.1.4 Brace Expansion

```
$ echo foo.{pdf,txt,png,jpg}
foo.pdf foo.txt foo.png foo.jpg
```

3.1.5 Variable Expansion

```
$ foo1="foo1"
$ foo2="foo2"

# expand to "$foo1 foo2"
$ echo "${!foo*}"

# expand to "$foo1" "$foo2"
$ echo "${!foo@}"
```

3.1.6 String length

```
echo ${#foo}
7
```

3.1.7 String Slice

```
$ foo="01234567890abcdefg"

# ${param:offset}
$ echo ${foo:7}
7890abcdefg

$ echo ${foo: -7}
abcdefg
$ echo ${foo: -7:2}
ab

# ${param:offset:length}
$ echo ${foo:7:3}
789
```

3.1.8 Delete Match String

```
$ foo="123,456,789"
# ${p##substring} delete longest match of substring from front
$ echo ${foo##*,}
789

# ${p#substring} delete shortest match of substring from front
echo ${foo#*,}
456,789

# ${p%%substring} delete longest match of substring from back
$ echo ${foo%%,*}
123
```

(continues on next page)

(continued from previous page)

```
$ echo ${foo%,*}
123,456
```

Other examples

```
disk="/dev/sda"
$ echo ${disk##*/}
sda

$ disk="/dev/sda3"
echo ${disk%%[0-9]*}
/dev/sda
```

3.1.9 Here Documents

```
cat <<EOF
    Hello Document
EOF
```

3.1.10 Here Strings

```
# CMD <<< $w, where $w is expanded to the stdin of CMD

bc <<< "1 + 2 * 3"
```

3.1.11 Logger

```
REST='\e[0m'
RED='\e[1;31m'
GREEN='\e[1;32m'
YELLOW='\e[1;33m'
CYAN='\e[1;36m'

info() {
  echo -e "[$(date +'%Y-%m-%dT%H:%M:%S%z')][${GREEN}info${REST}] $*"
}

debug() {
  echo -e "[$(date +'%Y-%m-%dT%H:%M:%S%z')][${CYAN}debug${REST}] $*"
}

warn() {
  echo -e "[$(date +'%Y-%m-%dT%H:%M:%S%z')][${YELLOW}warn${REST}] $*" >&2
}

err() {
  echo -e "[$(date +'%Y-%m-%dT%H:%M:%S%z')][${RED}error${REST}] $*" >&2
}
```

3.1.12 Check Command Exist

```
cmd="tput"
if command -v "${tput}" > /dev/null; then
  echo "$cmd exist"
else
  echo "$cmd does not exist"
fi
```

3.1.13 Read a File Line by Line

```
#!/bin/bash

file="file.txt"
while IFS= read -r l; do echo $l; done < "$file"
```

3.1.14 Read a File field wise

```
#!/bin/bash

file="/etc/passwd"
while IFS=: read -r n _ _ _ _ _ _; do echo $n; done < "$file"
```

3.1.15 Dictionary

```
#!/bin/bash

declare -A d
d=( ["foo"]="FOO" ["bar"]="BAR" )
d["baz"]="BAZ"

for k in "${!d[@]}"; do
  echo "${d[$k]}"
done
```

3.1.16 Check if a Key Exists in a Dictionary

```
#!/bin/bash

declare -A d
d["foo"]="FOO"
if [ -v "d[foo]" ]; then
  echo "foo exists in d"
else
  echo "foo does exists in d"
fi
```

3.1.17 Remove a Key-Value from a Dictionary

```
$ declare -A d
$ d["foo"]="FOO"
$ unset d["foo"]
```

3.1.18 Append Elements to an Array

```
#!/bin/bash

arr=()

for i in "a b c d e"; do
  arr+=($i)
done

echo "${arr[@]}"
```

3.1.19 Prompt

```
#!/bin/bash

read -p "Continue (y/n)? " c
case "$c" in
  y|Y|yes) echo "yes" ;;
  n|N|no) echo "no" ;;
  *) echo "invalid" ;;
esac
```

3.1.20 Parse Arguments

```
#!/bin/bash

program="$1"

usage() {
  cat <<EOF

Usage:  $program [OPTIONS] params

Options:

  -h,--help                show this help
  -a,--argument string     set an argument

EOF
}
```

(continues on next page)

(continued from previous page)

```
arg=""
params=""
while (( "$#" )); do
  case "$1" in
        -h|-\?|--help) usage; exit 0 ;;
        -a|--argument) args="$2"; shift 2 ;;
        # stop parsing
        --) shift; break ;;
        # unsupport options
        -*|--*=) echo "unsupported option $1" >&2; exit 1 ;;
        # positional arguments
        *) params="$params $1"; shift ;;
  esac
done
```

3.2 Bash Date cheatsheet

Table of Contents

3.2.1 Today

```
$ date
Sun Jun 20 15:23:20 CST 2021
$ date +"%Y%m%d"
20210620
```

3.2.2 N Days Before

```
# Linux
$ date +%Y%m%d -d "1 day ago"
20210619

# BSD (MacOS)
$ date -j -v-1d +"%Y%m%d"
20210619
```

3.3 Bash Find cheatsheet

Table of Contents

3.3.1 Find by Suffix

```
$ find "${path}" -name "*.py"
```

3.3.2 Find by Substring

```
$ find "${path}" -name "*code*"
```

3.3.3 Find by Case Insensitive

```
$ find "${path}" -iname "*.py"
```

3.3.4 Find by File Type

```
# b  block
# c  character
# d  directory
# p  named pipe
# f  regular file
# l  symbolic link
# s  socket

# find regular file
$ find "${path}" -type f -name "*.py"
```

(continues on next page)

(continued from previous page)

```
# find directory
$ find "${path}" -type d
```

3.3.5 Find by Size

```
# find files < 50M
$ find "${path}" -type f -size -50M

# find files > 50M
$ find "${path}" -type f -size +50M
```

3.3.6 Find by Date

```
# files are not accessed > 7 days
$ find "${path}" -type f -atime +7

# files are accessed < 7 days
$ find "${path}" -type f -atime -7

# files are not accessed > 10 min
$ find "${path}" -type f -amin +10

# files are accessed < 10 min
$ find "${path}" -type f -amin -10
```

3.3.7 Find by User

```
$ find "${path}" -type f -user "${USER}"
```

3.3.8 Delete after Find

```
# delete by pattern
$ find "${path}" -type f -name "*.sh" -delete

# delete recursively
find ker -type d -exec rm -rf {} \+
```

3.3.9 grep after find

```
$ find ker -type f -exec grep -rni "test" {} \+

# or

$ find ker -type f -exec grep -rni "test" {} \;
```

3.4 Bash Regular Expression Cheatsheet

Table of Contents

3.4.1 grep vs grep -E

The difference between grep and grep -E is that grep uses basic regular expressions while grep -E uses extended regular expressions. In basic regular expressions, the characters "?", "+", "{", "|", "(","")" lose their special meaning; instead, use "?", "+", "{", "|", "(", ")".

```
$ echo "987-123-4567" | grep "^[0-9]\{3\}-[0-9]\{3\}-[0-9]\{4\}$"
$ echo "987-123-4567" | grep -E "^[0-9]{3}-[0-9]{3}-[0-9]{4}$"
```

3.4.2 *tr* Substitutes Strings

```
# tr substitutes white spaces to newline
$ echo "a b c" | tr "[:space:]+" "\n"
a
b
c

# tr spueeze multiple spaces
$ echo "a    b   c" | tr -s " "
a b c
```

3.4.3 *uniq* Filters out Repeated Lines

```
$ echo "a a b b c" | tr " " "\n" | sort | uniq
a
b
c

# display count
$ echo "a a b b a c" | tr " " "\n" | sort | uniq -c
   3 a
   2 b
   1 c
```

Note that `uniq` only filters out lines continuously. However, if characters are equal but they does not appear continually, `uniq` does not squeeze them. Therefore, a programmer needs to use `sort` to categorizes lines before `uniq`.

```
$ echo "a a b b a c" | tr " " "\n" | uniq
a
b
a
c
```

3.4.4 `sort` lines

```
# sort by lines
$ echo "b a c d" | tr " " "\n" | sort

# sort by lines reversely
$ echo "b a c d" | tr " " "\n" | sort -r

# sort by a field
$ echo "b a b c d" | tr " " "\n" | sort | uniq -c | sort -k1
```

3.5 Operating System cheatsheet

Table of Contents

3.5.1 Get Number of CPUs

```
# linux
nproc --all

# max
sysctl -n hw.logicalcpu
```

CHAPTER

FOUR

SYSTEM PROGRAMMING CHEAT SHEET

4.1 C file operations cheatsheet

Table of Contents

4.1.1 Calculate file size via `lseek`

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    int fd = -1;
    size_t s_offset = 0;
    size_t e_offset = -1;
    char *path = NULL;
```

(continues on next page)

(continued from previous page)

```
    if (argc != 2) {
        printf("Usage: PROG file\n");
        goto Error;
    }
    path = argv[1];
    if(0 > (fd = open(path,O_RDONLY))) {
        printf("open failed\n");
        goto Error;
    }
    if (-1 == (s_offset = lseek(fd, 0, SEEK_SET))) {
        printf("lseek error\n");
        goto Error;
    }
    if (-1 == (e_offset = lseek(fd, 0, SEEK_END))) {
        printf("lseek error\n");
        goto Error;
    }
    printf("File Size: %ld byte\n", e_offset - s_offset);
    ret = 0;
Error:
    if (fd>=0) {
        close(fd);
    }
    return ret;
}
```

output:

```
$ echo "Hello" > hello.txt
$ ./a.out hello.txt
File Size: 6 byte
```

4.1.2 Using `fstat` get file size

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    int fd = -1;
    struct stat st = {0};
    char *path = NULL;

    if (argc != 2) {
        printf("Usage: PROG file\n");
        goto Error;
```

(continues on next page)

(continued from previous page)

```
    }
    path = argv[1];
    /* using fstat */
    if (-1 == (fd = open(path, O_RDONLY))) {
        printf("open file get error\n");
        goto Error;
    }
    if (-1 == fstat(fd, &st)) {
        printf("fstat get error\n");
        goto Error;
    }
    printf("File Size: %lld byte\n", st.st_size);
    ret = 0;
Error:
    if (fd>=0) {
        close(fd);
    }
    return ret;
}
```

output:

```
$ echo "Hello" > hello.txt
$ ./a.out hello.txt
File Size: 6 byte
```

4.1.3 Copy all content of a file

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

#define COPY_BUF_SIZE 1024

int main(int argc, char *argv[])
{
    int ret = -1;
    int sfd = -1, dfd = -1;
    mode_t perm = 0;
    char *src = NULL;
    char *dst = NULL;
    char buf[COPY_BUF_SIZE] = {0};
    size_t r_size = 0;
    struct stat st = {0};

    if (argc != 3) {
        printf("Usage: PROG src dst\n");
        goto Error;
```

(continues on next page)

(continued from previous page)

```
    }

    /* open source */
    src = argv[1];
    if (-1 == (sfd = open(src, O_RDONLY))) {
        printf("open source fail\n");
        goto Error;
    }
    /* read source permission */
    if (-1 == (fstat(sfd, &st))) {
        printf("fstat file error\n");
        goto Error;
    }
    /* copy destination */
    dst = argv[2];
    perm = st.st_mode; /* set file permission */
    if (-1 == (dfd = open(dst, O_WRONLY | O_CREAT, perm))) {
        printf("open destination fail\n");
        goto Error;
    }
    while (0 < (r_size = read(sfd, buf, COPY_BUF_SIZE))) {
        if (r_size != write(dfd, buf, r_size)) {
            printf("copy file get error\n");
            goto Error;
        }
    }
    ret = 0;
Error:
    if (sfd >= 0) {
        close(sfd);
    }
    if (dfd >= 0) {
        close(dfd);
    }
    return ret;
}
```

output:

```
$ echo "Hello" > hello.txt
$ ./a.out hello.txt hello_copy.txt
$ diff hello.txt hello_copy.txt
```

4.1.4 Copy some bytes of content to a file

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    int sfd = -1, dfd = -1;
    size_t s_offset = 0;
    size_t d_offset = -1;
    mode_t perm = 0;
    char *src = NULL;
    char *dst = NULL;
    struct stat st = {0};
    char buf[1024] = {0};
    size_t size = 0;
    size_t r_size = 0;

    if (argc != 4) {
        printf("Usage: PROG src dst bytes\n");
        goto Error;
    }
    /* open source file */
    src = argv[1];
    if (0 > (sfd = open(src, O_RDONLY))) {
        printf("open source file error\n");
        goto Error;
    }
    /* get source file permission */
    if (-1 == fstat(sfd, &st)) {
        printf("fstat fail\n");
        goto Error;
    }
    /* open dst file */
    dst = argv[2];
    perm = st.st_mode;
    if (0 > (dfd = open(dst, O_WRONLY | O_CREAT, perm))) {
        printf("open destination file error\n");
        goto Error;
    }
    if (-1 == (d_offset = lseek(dfd, 0, SEEK_END))) {
        printf("lseek get error\n");
        goto Error;
    }
    if (-1 == (s_offset = lseek(sfd, d_offset, SEEK_SET))) {
        printf("lseek get error\n");
        goto Error;
    }
```

(continues on next page)

(continued from previous page)

```
    /* bytes */
    size = atoi(argv[3]);
    if (-1 == (r_size = read(sfd, buf, size))) {
        printf("read content fail\n");
        goto Error;
    }
    if (r_size != write(dfd, buf, r_size)) {
        printf("write content fail\n");
        goto Error;
    }
    ret = 0;
Error:
    if (sfd >= 0) {
        close(sfd);
    }
    if (dfd >= 0) {
        close(dfd);
    }
    return ret;
}
```

output:

```
$ echo "Hello" > hello.txt
$ $ ./a.out hello.txt hello_copy.txt 3
$ cat hello_copy.txt
Hel$./a.out hello.txt hello_copy.txt 3
$ cat hello_copy.txt
Hello
$ diff hello.txt hello_copy.txt
```

4.1.5 Get lines of a file

```
// basic API: fopen, getline

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    FILE *f = NULL;
    ssize_t read_size = 0;
    size_t len = 0;
    char *path = NULL;
    char *line = NULL;

    if (argc != 2) {
        printf("Usage: PROG file\n");
        goto Error;
```

(continues on next page)

(continued from previous page)

```
    }

    path = argv[1];
    if (NULL == (f = fopen(path, "r"))) {
        printf("Read file error");
        goto Error;
    }

    while (-1 != getline(&line, &len, f)) {
        printf("%s\n", line);
    }
    ret = 0;
Error:
    if (line) {
        free(line);
        line = NULL;
    }
    if (f) {
        fclose(f);
    }
    return ret;
}
```

4.1.6 Get lines of a file via `std::getline`

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>

int main(int argc, char *argv[])
{
    std::ifstream f(argv[1]);
    for (std::string line; std::getline(f, line);) {
        std::cout << line << "\n";
    }
}
```

4.1.7 Read content into memory from a file

```
// basick API: fopen, fseek, ftell, rewind, fread
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    FILE *f = NULL;
    char *path = NULL;
```

(continues on next page)

(continued from previous page)

```
    int size = 0;
    int read_size = 0;
    char *buffer = NULL;

    if (argc != 2) {
        printf("Usage: PROG file\n");
        goto Error;
    }

    path = argv[1];
    if (NULL == (f = fopen(path, "r"))) {
        printf("Read %s into memory fail\n", path);
        goto Error;
    }
    fseek(f, 0, SEEK_END);
    size = ftell(f);
    rewind(f);

    if (NULL == (buffer = (char *)calloc(size, sizeof(char)))) {
        printf("malloc file fail\n");
        goto Error;
    }

    read_size = fread(buffer, 1, size, f);
    if (read_size != size) {
        printf("fread %s fail\n", path);
        goto Error;
    }
    buffer[size-1] = '\0';
    printf("%s\n", buffer);
    ret = 0;
Error:
    if (buffer) {
        free(buffer);
        buffer = NULL;
    }
    if (f) {
        fclose(f);
    }
    return ret;
}
```

4.1.8 Check file types

```c
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    struct stat st;
    char *path = NULL;

    bzero(&st, sizeof(struct stat));

    if (argc != 2) {
        printf("Usage: PROG file\n");
        goto Error;
    }
    path = argv[1];
    if (-1 == stat(path, &st)) {
        printf("stat %s get error\n", path);
        goto Error;
    }
    /* check file type */
    switch (st.st_mode & S_IFMT) {
        case S_IFBLK: printf("Block device\n"); break;
        case S_IFCHR: printf("Character device\n"); break;
        case S_IFDIR: printf("Directory\n"); break;
        case S_IFIFO: printf("FIFO pipe\n"); break;
        case S_IFLNK: printf("Symbolic link\n"); break;
        case S_IFREG: printf("Regular file\n"); break;
        case S_IFSOCK: printf("Socket\n"); break;
        default: printf("Unknown\n");
    }
    ret = 0;
Error:
    return ret;
}
```

output:

```
$ ./a.out /etc/hosts
Regular file
$ ./a.out /usr
Directory
./a.out /dev/tty.Bluetooth-Incoming-Port
Character device
```

4.1.9 File tree walk

```c
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <ftw.h>

#define CHECK_RET(ret, fmt, ...)        \
    do {                                \
        if (ret < 0) {                  \
            printf(fmt, ##__VA_ARGS__); \
            goto End;                   \
        }                               \
    } while(0)

#define CHECK_NULL(ret, fmt, ...)       \
    do {                                \
        if (ret == NULL) {              \
            printf(fmt, ##__VA_ARGS__); \
            goto End;                   \
        }                               \
    } while(0)

int callback(const char *fpath, const struct stat *sb, int typeflag, struct FTW *ftwbuf)
{
    CHECK_NULL(fpath, "fpath cannot be NULL\n");
    printf("%s\n", fpath);
End:
    return 0;
}

int main(int argc, char *argv[])
{
    int ret = -1;
    char *path = NULL;

    if (argc != 2) {
        perror("Usage: PROG [dirpath]\n");
        goto End;
    }

    path = argv[1];
    ret = nftw(path, callback, 64, FTW_DEPTH | FTW_PHYS);
    CHECK_RET(ret, "nftw(%s) fail. [%s]", path, strerror(errno));
End:
    return ret;
}
```

output:

```
$ gcc tree_walk.c
```

(continues on next page)

(continued from previous page)

```
$ ./a.out .
./tree_walk.c
./a.out
.
```

4.2 C signal operation cheatsheet

Table of Contents

4.2.1 Print signal expression

```c
#include <stdio.h>
#include <signal.h>

#define ARRAYLEN(arr) sizeof(arr) / sizeof((arr)[0])

static int signo_arr[] = {
    SIGABRT , SIGALRM  , SIGBUS,
    SIGCHLD , SIGCONT  , SIGFPE,
    SIGHUP  , SIGILL   , SIGINT,
    SIGIO   , SIGKILL  , SIGPIPE,
    SIGPROF , SIGQUIT  , SIGSEGV,
    SIGSYS  , SIGTERM  , SIGTRAP,
    SIGTSTP , SIGTTIN  , SIGTTOU,
    SIGURG  , SIGVTALRM, SIGUSR1,
    SIGUSR2 , SIGXCPU  , SIGXFSZ
};

int main(int argc, char *argv[])
{
    int i = 0;
    int signo = -1;
    char *msg = "SIGNAL";

    for (i=0; i < ARRAYLEN(signo_arr); i++) {
        signo = signo_arr[i];
```

(continues on next page)

(continued from previous page)

```
        printf("Signal[%d]: %s\n", signo, sys_siglist[signo]);
    }

    return 0;
}
```

output:

```
$ ./a.out
Signal[6]: Abort trap
Signal[14]: Alarm clock
Signal[10]: Bus error
Signal[20]: Child exited
Signal[19]: Continued
Signal[8]: Floating point exception
Signal[1]: Hangup
Signal[4]: Illegal instruction
Signal[2]: Interrupt
Signal[23]: I/O possible
Signal[9]: Killed
Signal[13]: Broken pipe
Signal[27]: Profiling timer expired
Signal[3]: Quit
Signal[11]: Segmentation fault
Signal[12]: Bad system call
Signal[15]: Terminated
Signal[5]: Trace/BPT trap
Signal[18]: Suspended
Signal[21]: Stopped (tty input)
Signal[22]: Stopped (tty output)
Signal[16]: Urgent I/O condition
Signal[26]: Virtual timer expired
Signal[30]: User defined signal 1
Signal[31]: User defined signal 2
Signal[24]: Cputime limit exceeded
Signal[25]: Filesize limit exceeded
```

4.2.2 Basic signal event handler

```
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>

/** singal handler prototype :
 *
 *  type void (*sighandler_t) (int)
 */
```

(continues on next page)

(continued from previous page)

```
void sig_handler(int signo)
{
    printf("[%d] Get signal: %s\n", getpid(), strsignal(signo));
}

int main(int argc, char *argv[])
{
    int ret = -1;

    /* overwrite default signal handler */
    if (SIG_ERR == signal(SIGHUP, sig_handler)) {
        printf("Get error: %s\n", strerror(errno));
        goto Error;
    }
    if (SIG_ERR == signal(SIGINT, sig_handler)) {
        printf("Get error: %s\n", strerror(errno));
        goto Error;
    }
    if (SIG_ERR == signal(SIGALRM, sig_handler)) {
        printf("Get error: %s\n", strerror(errno));
        goto Error;
    }
    /* ignore signal */
    if (SIG_ERR == signal(SIGUSR1, SIG_IGN)) {
        printf("Get error: %s\n", strerror(errno));
        goto Error;
    }
    while(1) { sleep(3); }
    ret = 0;
Error:
    return ret;
}
```

output:

```
$ ./a.out
^C[54652] Get signal: Interrupt: 2
[54652] Get signal: Hangup: 1
[54652] Get signal: Alarm clock: 14
```

4.2.3 A pthread signal handler

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>
```

(continues on next page)

(continued from previous page)

```
static void *sig_thread(void *arg)
{
    sigset_t *set = (sigset_t *)arg;
    int err = -1, signo = -1;

    for(;;) {
        if(0 != (err = sigwait(set, &signo))) {
            printf("sigwait error\n");
            goto Error;
        }
        printf("Get signal[%d]: %s\n",
               signo, sys_siglist[signo]);
    }
Error:
    return;
}

int main(int argc, char *argv[])
{
    pthread_t thread;
    sigset_t sig_set;
    int err = -1;

    sigemptyset(&sig_set);
    sigaddset(&sig_set, SIGQUIT);
    sigaddset(&sig_set, SIGUSR1);
    /* set signal handler thread sigmask */
    err = pthread_sigmask(SIG_BLOCK, &sig_set, NULL)
    if(0 != err) {
        printf("set pthread_sigmask error\n");
        goto Error;
    }
    /* create signal thread */
    err = pthread_create(&thread, NULL,
                         &sig_thread, (void *)&sig_set))
    if (0 != err) {
        printf("create pthread error\n");
        goto Error;
    }

    pause();
Error:
    return err;
}
```

output:

```
$ ./a.out &
[1] 21258
$ kill -USR1 %1
Get signal[10]: User defined signal 1
$ kill -QUIT %1
```

(continues on next page)

(continued from previous page)

```
Get signal[3]: Quit
$ kill -TERM %1
[1]+  Terminated              ./a.out
```

4.2.4 Check child process alive

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

void handler(int signo)
{
    pid_t pid = getpid();
    printf("[%i] Got signal[%d]: %s\n",
           pid, signo, sys_siglist[signo]);
}

int main(int argc, char *argv[])
{
    int ret = -1;
    pid_t pid = -1;

    pid = fork();
    signal(SIGCHLD, handler);
    if (pid < 0) {
        printf("Fork failed\n");
        goto Error;
    } else if (pid == 0) {
        /* child */
        printf("Child[%i]\n", getpid());
        sleep(3);
    } else {
        printf("Parent[%i]\n", getpid());
        pause();
    }
    ret = 0;
Error:
    return ret;
}
```

```
$ ./a.out
Parent[59113]
Child[59114]
[59113] Got signal[20]: Child exited
```

4.2.5 Basic sigaction usage

```c
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <unistd.h>

void handler(int signo)
{
    printf("Get Signal: %s\n",sys_siglist[signo]);
}

int main(int argc, char *argv[])
{
    pid_t pid = -1;
    struct sigaction new_sa = {0};
    struct sigaction old_sa = {0};

    new_sa.sa_handler = handler;
    sigemptyset(&new_sa.sa_mask);
    new_sa.sa_flags = 0;

    pid = getpid();
    printf("Process PID: %i\n", pid);
    /* if signal not ignore, overwrite its handler */
    sigaction(SIGINT, NULL, &old_sa);
    if (old_sa.sa_handler != SIG_IGN) {
        sigaction(SIGINT, &new_sa, NULL);
    }

    sigaction(SIGHUP, NULL, &old_sa);
    if (old_sa.sa_handler != SIG_IGN) {
        sigaction(SIGHUP, &new_sa, NULL);
    }
    while (1) { sleep(3); }
    return 0;
}
```

output:

```
# bash 1
kill -1 57140
kill -2 57140

# bash 2
$ ./a.out
Process PID: 57140
Get Signal: Hangup
Get Signal: Interrupt
```

4.2.6 Block & Unblock signal

```c
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <signal.h>
#include <setjmp.h>

static sigjmp_buf jmpbuf;

void handler(int signo)
{
    printf("Get signal[%d]: %s\n", signo, sys_siglist[signo]);
    if (SIGUSR1 == signo) {
        siglongjmp(jmpbuf, 1);
    }
}

int main(int argc, char *argv[])
{
    int ret = -1;
    sigset_t new_mask, old_mask;

    sigemptyset(&new_mask);
    sigaddset(&new_mask, SIGHUP);

    if (SIG_ERR == signal(SIGHUP, handler)) {
            printf("Set signal get %s error", strerror(errno));
            goto Error;
    }
    if (SIG_ERR == signal(SIGALRM, handler)) {
            printf("Set signal get %s error", strerror(errno));
            goto Error;
    }
    if (SIG_ERR == signal(SIGUSR1, handler)) {
            printf("Set signal get %s error", strerror(errno));
            goto Error;
    }
    /* block SIGHUP */
    if (sigsetjmp(jmpbuf, 1)) {
            /* unblock SIGHUP */
            sigprocmask(SIG_UNBLOCK, &new_mask, &old_mask);
    } else {
            /* block SIGHUP */
            sigprocmask(SIG_BLOCK, &new_mask, &old_mask);
    }
    while (1) sleep(3);
    ret = 0;
Error:
    return ret;
}
```

output:

```
$ kill -HUP %1
$ kill -ALRM %1
Get signal[14]: Alarm clock
$ kill -USR1 %1
Get signal[10]: User defined signal 1
Get signal[1]: Hangup
```

4.3 C socket cheatsheet

Table of Contents

4.3.1 Get host via `gethostbyname`

```c
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int ret = -1, i = 0;
    struct hostent *h_ent = NULL;
    struct in_addr **addr_list = NULL;

    if (argc != 2) {
        printf("Usage: COMMAND [name]\n");
        goto End;
    }
    h_ent = gethostbyname(argv[1]);
    if (h_ent == NULL) {
        printf("gethostbyname fail. %s", hstrerror(h_errno));
        goto End;
    }
```

(continues on next page)

(continued from previous page)

```
    printf("Host Name: %s\n", h_ent->h_name);
    addr_list = (struct in_addr **)h_ent->h_addr_list;
    for (i=0; addr_list[i] != NULL; i++) {
        printf("IP Address: %s\n", inet_ntoa(*addr_list[i]));
    }

    ret = 0;
End:
    return ret;
}
```

output:

```
$ cc -g -Wall -o gethostbyname gethostbyname.c
$ ./gethostbyname localhost
Host Name: localhost
IP Address: 127.0.0.1
$ ./gethostbyname www.google.com
Host Name: www.google.com
IP Address: 74.125.204.99
IP Address: 74.125.204.105
IP Address: 74.125.204.147
IP Address: 74.125.204.106
IP Address: 74.125.204.104
IP Address: 74.125.204.103
```

4.3.2 Transform host & network endian

```
#include <stdio.h>
#include <stdint.h>
#include <arpa/inet.h>

static union {
    uint8_t buf[2];
    uint16_t uint16;
} endian = { {0x00, 0x3a} };

#define LITTLE_ENDIANNESS ((char)endian.uint16 == 0x00)
#define BIG_ENDIANNESS ((char)endian.uint16 == 0x3a)

int main(int argc, char *argv[])
{
    uint16_t host_short_val = 0x01;
    uint16_t net_short_val = 0;
    uint32_t host_long_val = 0x02;
    uint32_t net_long_val = 0;

    net_short_val  = htons(host_short_val);
    net_long_val   = htonl(host_long_val);
    host_short_val = htons(net_short_val);
```

(continues on next page)

(continued from previous page)

```
    host_long_val  = htonl(net_long_val);

    if (LITTLE_ENDIANNESS) {
        printf("On Little Endian Machine:\n");
    } else {
        printf("On Big Endian Machine\n");
    }
    printf("htons(0x%x) = 0x%x\n", host_short_val, net_short_val);
    printf("htonl(0x%x) = 0x%x\n", host_long_val, net_long_val);

    host_short_val = htons(net_short_val);
    host_long_val  = htonl(net_long_val);

    printf("ntohs(0x%x) = 0x%x\n", net_short_val, host_short_val);
    printf("ntohl(0x%x) = 0x%x\n", net_long_val, host_long_val);
    return 0;
}
```

output:

```
# on little endian machine
$ ./a.out
On Little Endian Machine:
htons(0x1) = 0x100
htonl(0x2) = 0x2000000
ntohs(0x100) = 0x1
ntohl(0x2000000) = 0x2

# on big endian machine
$ ./a.out
On Big Endian Machine
htons(0x1) = 0x1
htonl(0x2) = 0x2
ntohs(0x1) = 0x1
ntohl(0x2) = 0x2
```

4.3.3 Basic TCP socket server

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define BUF_SIZE 1024
#define isvalidsock(s) (s > 0 ? 1 : 0 )

static int port = 5566;

int main(int argc, char *argv[])
```

(continues on next page)

(continued from previous page)

```
{
    int ret = -1;
    int s = -1;
    int c = -1;
    socklen_t clen = 0;
    ssize_t len = 0;
    struct sockaddr_in s_addr;
    struct sockaddr_in c_addr;
    const int on = 1;
    char buf[BUF_SIZE] = {0};

    /* set socket host and port */
    bzero(&s_addr, sizeof(s_addr));
    s_addr.sin_family = AF_INET;
    s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    s_addr.sin_port = htons(port);

    /* create socket */
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (!isvalidsock(s)) {
        printf("Create socket fail\n");
        goto Error;
    }
    /* set sockopt */
    if (0 > setsockopt(s, SOL_SOCKET,
            SO_REUSEADDR, &on, sizeof(on))) {
        printf("setsockopt fail\n");
        goto Error;
    }
    /* bind address and port */
    if (0 > bind(s, (struct sockaddr *) &s_addr,
            sizeof(s_addr))) {
        printf("bind socket fail\n");
        goto Error;
    }
    /* listen */
    if (0 > listen(s, 10)) {
        printf("listen fail\n");
        goto Error;
    }
    for(;;) {
        clen = sizeof(c_addr);
        c = accept(s, (struct sockaddr *)&c_addr, &clen);
        if (!isvalidsock(c)) {
            printf("accept error\n");
            continue;
        }
        bzero(buf, BUF_SIZE);
        if (0 > (len = recv(c, buf, BUF_SIZE-1, 0))) {
            close(c);
        }
        send(c, buf, BUF_SIZE-1, 0);
```

(continues on next page)

(continued from previous page)

```
        close(c);
    }
    ret = 0
Error:
    if (s >= 0){
        close(s);
    }
    return ret;
}
```

output:

```
$ ./a.out &
[1] 63546
$ nc localhost 5566
Hello Socket
Hello Socket
```

4.3.4 Basic UDP socket server

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <unistd.h>

#define EXPECT_GE(i, e, ...) \
  if (i < e) {__VA_ARGS__}

#define EXPECT_SUCCESS(ret, fmt, ...) \
  EXPECT_GE(ret, 0, \
    printf(fmt, ##__VA_ARGS__); goto End;)

#ifndef BUF_SIZE
#define BUF_SIZE 1024
#endif

int main(int argc, char *argv[])
{
    int ret = -1;
    int sockfd = -1;
    int port = 5566;
    char buf[BUF_SIZE] = {};
    struct sockaddr_in s_addr = {};
    struct sockaddr_in c_addr = {};
    socklen_t s_len = 0;
```

(continues on next page)

(continued from previous page)

```
    /* create socket */
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    EXPECT_SUCCESS(sockfd, "create socket fail. %s\n", strerror(errno));


    /* set socket addr */
    bzero((char *) &s_addr, sizeof(s_addr));
    s_addr.sin_family = AF_INET;
    s_addr.sin_port = htons(port);
    s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    s_len = sizeof(c_addr);

    /* bind */
    ret = bind(sockfd, (struct sockaddr *)&s_addr, sizeof(s_addr));
    EXPECT_SUCCESS(ret, "bind fail. %s\n", strerror(errno));

    for(;;) {
        bzero(buf, sizeof(buf));
        ret = recvfrom(sockfd, buf, sizeof(buf), 0,
                       (struct sockaddr *)&c_addr, &s_len);
        EXPECT_GE(ret, 0, continue;);

        ret = sendto(sockfd, buf, ret, 0,
                     (struct sockaddr *)&c_addr, s_len);
    }

    ret = 0;
End:
    if (sockfd >= 0) {
        close(sockfd);
    }
    return ret;
}
```

output:

```
$ cc -g -Wall -o udp_server udp_server.c
$ ./udp_server &
[1] 90190
$ nc -u 192.168.55.66 5566
Hello
Hello
UDP
UDP
```

4.3.5 Event driven socket via `select`

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>

#define BUF_SIZE 1024
#define isvalidsock(s) (s > 0 ? 1 : 0)
#define PORT 5566

int socket_init(void)
{
    struct sockaddr_in s_addr;
    int sfd = -1;
    int ret = -1;
    const int on = 1;

    bzero(&s_addr, sizeof(s_addr));
    s_addr.sin_family = AF_INET;
    s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    s_addr.sin_port = htons(PORT);

    sfd = socket(AF_INET, SOCK_STREAM, 0);
    if (!isvalidsock(sfd)) {
        printf("create socket error\n");
        goto Error;
    }
    if (0 > setsockopt(
            sfd, SOL_SOCKET,
            SO_REUSEADDR, &on, sizeof(on))) {
        printf("setsockopt error\n");
        goto Error;
    }
    if (0 > bind(sfd,
                (struct sockaddr *)&s_addr,
                sizeof(s_addr))) {
        printf("bind error\n");
        goto Error;
    }
    if (0 > listen(sfd, 10)) {
        printf("listen network error\n");
        goto Error;
    }
    ret = sfd;
Error:
    if (ret == -1) {
        if (sfd >=0) {
            close(sfd);
        }
```

(continues on next page)

(continued from previous page)

```
    }
    return ret;
}

int main(int argc, char *argv[])
{
    int ret = -1;
    int sfd = -1;
    int cfd = -1;
    ssize_t len = 0;
    struct sockaddr_in c_addr;
    int i = 0;
    int rlen = 0;
    char buf[BUF_SIZE] = {0};
    socklen_t clen = 0;
    fd_set wait_set;
    fd_set read_set;

    if (-1 == (sfd = socket_init())) {
        printf("socket_init error\n");
        goto Error;
    }
    FD_ZERO(&wait_set);
    FD_SET(sfd, &wait_set);
    for (;;) {
        read_set = wait_set;
        if (0 > select(FD_SETSIZE, &read_set,
                       NULL, NULL, NULL)) {
            printf("select get error\n");
            goto Error;
        }
        for (i=0; i < FD_SETSIZE; i++) {
            if (!FD_ISSET(i, &read_set)) {
                continue;
            }
            if (i == sfd) {
                clen = sizeof(c_addr);
                cfd = accept(sfd,
                    (struct sockaddr *)&c_addr, &clen);
                if (!isvalidsock(cfd)) {
                    goto Error;
                }
                FD_SET(cfd, &wait_set);
            } else {
                bzero(buf, BUF_SIZE);
                if (0 > (rlen = read(i, buf, BUF_SIZE-1))) {
                    close(i);
                    FD_CLR (i, &wait_set);
                    continue;
                }
                if (0 > (rlen = write(i, buf, BUF_SIZE-1))) {
                    close(i);
```

(continues on next page)

(continued from previous page)

```
                    FD_CLR (i, &wait_set);
                    continue;
                }
            }
        }
    }
    ret = 0;
Error:
    if (sfd >= 0) {
        FD_CLR(sfd, &wait_set);
        close(sfd);
    }
    return ret;
}
```

output: (bash 1)

```
$ ./a.out &
[1] 64882
Hello
Hello
```

output: (bash 2)

```
$ nc localhost 5566
Socket
Socket
```

4.3.6 socket with pthread

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <pthread.h>

#define EXPECT_GE(i, e, ...) \
    if (i < e) { __VA_ARGS__; }

#define EXPECT_SUCCESS(ret, fmt, ...) \
    EXPECT_GE(ret, 0, printf(fmt, ##__VA_ARGS__); goto End)

#define SOCKET(sockfd, domain, types, proto) \
    do { \
        sockfd = socket(domain, types, proto); \
        EXPECT_SUCCESS(sockfd, "create socket fail. %s", strerror(errno)); \
    } while(0)
```

(continues on next page)

(continued from previous page)

```
#define SETSOCKOPT(ret, sockfd, level, optname, optval) \
    do { \
        int opt = optval;\
        ret = setsockopt(sockfd, level, optname, &opt, sizeof(opt)); \
        EXPECT_SUCCESS(ret, "setsockopt fail. %s", strerror(errno)); \
    } while(0)

#define BIND(ret, sockfd, addr, port) \
    do { \
        struct sockaddr_in s_addr = {}; \
        struct sockaddr sa = {}; \
        socklen_t len = 0; \
        ret = getsockname(sockfd, &sa, &len); \
        EXPECT_SUCCESS(ret, "getsockopt fail. %s", strerror(errno)); \
        s_addr.sin_family = sa.sa_family; \
        s_addr.sin_addr.s_addr = inet_addr(addr); \
        s_addr.sin_port = htons(port); \
        ret = bind(sockfd, (struct sockaddr *) &s_addr, sizeof(s_addr)); \
        EXPECT_SUCCESS(ret, "bind fail. %s", strerror(errno)); \
    } while(0)

#define LISTEN(ret, sockfd, backlog) \
    do { \
        ret = listen(sockfd, backlog); \
        EXPECT_SUCCESS(ret, "listen fail. %s", strerror(errno)); \
    } while(0)


#ifndef BUF_SIZE
#define BUF_SIZE 1024
#endif

void *handler(void *p_sockfd)
{
    int ret = -1;
    char buf[BUF_SIZE] = {};
    int c_sockfd = *(int *)p_sockfd;

    for (;;) {
        bzero(buf, sizeof(buf));
        ret = recv(c_sockfd, buf, sizeof(buf) - 1, 0);
        EXPECT_GE(ret, 0, break);
        send(c_sockfd, buf, sizeof(buf) - 1, 0);
    }
    EXPECT_GE(c_sockfd, 0, close(c_sockfd));
    pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
    int ret = -1, sockfd = -1, c_sockfd = -1;
```

(continues on next page)

(continued from previous page)

```
    int port = 9527;
    char addr[] = "127.0.0.1";
    struct sockaddr_in c_addr = {};
    socklen_t clen = 0;
    pthread_t t;

    SOCKET(sockfd, AF_INET, SOCK_STREAM, 0);
    SETSOCKOPT(ret, sockfd, SOL_SOCKET, SO_REUSEADDR, 1);
    BIND(ret, sockfd, addr, port);
    LISTEN(ret, sockfd, 10);

    for(;;) {
        c_sockfd = accept(sockfd, (struct sockaddr *)&c_addr, &clen);
        EXPECT_GE(c_sockfd, 0, continue);
        ret = pthread_create(&t, NULL, handler, (void *)&c_sockfd);
        EXPECT_GE(ret, 0, close(c_sockfd); continue);
    }
End:
    EXPECT_GE(sockfd, 0, close(sockfd));
    ret = 0;
    return ret;
}
```

output:

```
# console 1
$ cc -g -Wall -c -o test.o test.c
$ cc test.o -o test
$ ./test &
[1] 86601
$ nc localhost 9527
Hello
Hello

# console 2
$ nc localhost 9527
World
World
```

4.4 C Concurrency cheatsheet

Table of Contents

4.4.1 How to write a UNIX daemon

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <sys/stat.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    pid_t pid;

    /* become daemon */

    pid = fork();
    if (-1 == pid) {
        printf("Fork get error\n");
        goto Error;
    } else if (pid != 0) {
        ret = 0;
        goto Error;
    }
    /* Change the file mode mask */
    umask(0);

    /* set sid */
    if (-1 == setsid()) {
        printf("set sid failed\n");
        goto Error;
    }
    /* chdir to root "/" */
    if (-1 == chdir("/")) {
        printf("chdir(\"/\") failed\n");
        goto Error;
    }
    /* close stdin, stdout, stderr */
    close(STDIN_FILENO);
    close(STDOUT_FILENO);
    close(STDERR_FILENO);

    /* Do some task here */
    while (1) { sleep(3); syslog(LOG_ERR, "Hello"); }

    ret = 0;
Error:
    return ret;
}
```

4.4.2 Using daemon(nochdir, noclose)

```c
#include <stdio.h>
#include <unistd.h>
#include <syslog.h>

int main(int argc, char *argv[])
{
    int ret = -1;
    /* make process as a daemon */
    if (-1 == daemon(0, 0)) {
        syslog(LOG_ERR, "create a daemon get error");
        goto Error;
    }
    /* do the daemon task */
    while(1) { sleep(3); syslog(LOG_ERR, "Hello"); }
    ret = 0;
Error:
    return ret;
}
```

CHAPTER

FIVE

CMAKE CHEAT SHEET

5.1 cmake

Table of Contents

5.1.1 Minimal CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)
add_executable(a.out a.cpp b.cpp)
```

5.1.2 Wildcard Sourse Files

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")

project(example)
add_executable(a.out ${src})
```

5.1.3 Set CXXFLAGS

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cc")

project(example)
set(CMAKE_CXX_FLAGS "-Wall -Werror -O3")
add_executable(a.out ${src})
```

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cc")

project(example)
add_executable(a.out ${src})
target_compile_options(a.out PRIVATE -Werror)
```

5.1.4 Set CXXFLAGS with Build Type

```
# common
set(CMAKE_CXX_FLAGS "-Wall -Werror -O3")
# debug
set(CMAKE_CXX_FLAGS_DEBUG "${CMAKE_CXX_FLAGS} -g")
# release
set(CMAKE_C_FLAGS_RELEASE "${CMAKE_CXX_FLAGS} -O3 -pedantic")
```

5.1.5 Build Debug/Release

```
$ cmake -DCMAKE_BUILD_TYPE=Release ../
$ cmake -DCMAKE_BUILD_TYPE=Debug ../
```

5.1.6 Build with Type

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)
add_executable(a.out a.cc)

if(CMAKE_BUILD_TYPE STREQUAL "Debug")
  target_compile_options(a.out PRIVATE -g -O0 -Wall)
else()
  target_compile_options(a.out PRIVATE -O3 -Wall -Werror)
endif()
```

Instead of checking `CMAKE_BUILD_TYPE`, modern CMake prefers to use *generator expressions* to examine conditions.

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)
add_executable(a.out a.cc)
target_compile_options(a.out PRIVATE
  $<IF:$<CONFIG:Debug>, -g -O0 -Wall, -O3 -Wall -Werror>
)
```

5.1.7 Version File

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")

project(example VERSION 1.0)
configure_file(version.h.in version.h)

add_executable(a.out ${src})
target_include_directories(a.out PUBLIC "${PROJECT_BINARY_DIR}")
```

version.h.in

```
#pragma once

#define VERSION_MAJOR @example_VERSION_MAJOR@
#define VERSION_MINOR @example_VERSION_MINOR@
```

5.1.8 Build/Link a static library

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")

project(example VERSION 1.0)
configure_file(version.h.in version.h)

add_executable(a.out ${src})
add_library(b b.cpp)
target_link_libraries(a.out PUBLIC b)
target_include_directories(a.out PUBLIC "${PROJECT_BINARY_DIR}")
```

5.1.9 Build/Link a shared library

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")

project(example VERSION 1.0)
configure_file(version.h.in version.h)

add_executable(a.out ${src})
add_library(b SHARED b.cpp)
target_link_libraries(a.out PUBLIC b)
target_include_directories(a.out PUBLIC "${PROJECT_BINARY_DIR}")
```

5.1.10 Subdirectory

subdirectory fib/

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")
add_library(b SHARED b.cpp)
target_include_directories(b PUBLIC "${CMAKE_CURRENT_SOURCE_DIR}")
```

project dir

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
file(GLOB src "*.cpp")

project(example VERSION 1.0)
configure_file(version.h.in version.h)
```

(continues on next page)

(continued from previous page)

```
add_executable(a.out ${src})
add_subdirectory(fib)
target_link_libraries(a.out PUBLIC b)
target_include_directories(a.out PUBLIC
    "${PROJECT_BINARY_DIR}"
    "${PROJECT_BINARY_DIR/fib}"
)
```

5.1.11 PUBLIC & PRIVATE

- PUBLIC - only affect the current target, not dependencies
- INTERFACE - only needed for dependencies

```
cmake_minimum_required(VERSION 3.10)

project(example)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
find_package(Boost)

add_executable(a.out a.cpp)
add_library(b STATIC b.cpp b.h)

target_include_directories(a.out PRIVATE "${CMAKE_CURRENT_SOURCE_DIR}")
target_include_directories(b PRIVATE "${Boost_INCLUDE_DIR}")
target_link_libraries(a.out INTERFACE b) # link b failed
```

5.1.12 Generator Expression

```
cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)

set(target fib)
add_library(${target} src/fib.cc)
target_compile_options(${target} PRIVATE -Wall -Werror -Wextra)
target_include_directories(${target}
  PUBLIC
    $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
    $<INSTALL_INTERFACE:include>
  PRIVATE
    ${CMAKE_CURRENT_SOURCE_DIR}/src
)
```

5.1.13 Install

```
cmake_minimum_required(VERSION 3.10)
project(a)
add_library(b_static STATIC b.cc)
add_library(b_shared SHARED b.cc)
add_executable(a a.cc b.cc)

include(GNUInstallDirs)
set(INSTALL_TARGETS a b_static b_shared)
install(TARGETS ${INSTALL_TARGETS}
  ARCHIVE DESTINATION ${CMAKE_INSTALL_LIBDIR}
  LIBRARY DESTINATION ${CMAKE_INSTALL_LIBDIR}
  RUNTIME DESTINATION ${CMAKE_INSTALL_BINDIR}
)
install(FILES b.h DESTINATION ${CMAKE_INSTALL_INCLUDEDIR})
```

5.1.14 Run a command at configure time

```
execute_process(
    COMMAND git submodule update --init --recursive
    WORKING_DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR}
    RESULT_VARIABLE GIT_SUBMOD_RESULT
)
```

5.1.15 Option

```
# $ make -p build
# $ cd build
# $ cmake -DBUILD_TEST=ON ../

option(BUILD_TEST "Build test" OFF)
if (BUILD_TEST)
    message("Build tests.")
else()
    message("Ignore tests.")
endif()
```

5.1.16 Alias a Library

When a `CMakeLists.txt` export Foo in namespace `Foo::`, it also need to create an alias `Foo::Foo`.

```
add_library(Foo::Foo ALIAS Foo)
```

5.2 Package

Table of Contents

5.2.1 Support `find_package`

```
# fib
# ├── CMakeLists.txt
# ├── cmake
# │   └── FibConfig.cmake
# ├── include
# │   └── fib
# │       └── fib.h
# └── src
#     └── fib.cc

cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(example)

set(target fib)
add_library(${target} src/fib.cc)
target_compile_options(${target} PRIVATE -Wall -Werror -Wextra)
target_include_directories(${target}
  PUBLIC
    $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
    $<INSTALL_INTERFACE:include>
  PRIVATE
    ${CMAKE_CURRENT_SOURCE_DIR}/src
)

include(GNUInstallDirs)
install(TARGETS ${target} EXPORT FibTargets
  LIBRARY DESTINATION lib
  ARCHIVE DESTINATION lib
  RUNTIME DESTINATION bin
  INCLUDES DESTINATION include
)

install(DIRECTORY include/fib
  DESTINATION ${CMAKE_INSTALL_INCLUDEDIR}
)
install(EXPORT FibTargets
  FILE FibTargets.cmake
```

(continues on next page)

(continued from previous page)

```
  NAMESPACE Fib::
  DESTINATION lib/cmake/Fib
)
include(CMakePackageConfigHelpers)
write_basic_package_version_file("FibConfigVersion.cmake"
  VERSION 1.0.0
  COMPATIBILITY SameMajorVersion
)
install(FILES "cmake/FibConfig.cmake"
  "${PROJECT_BINARY_DIR}/FibConfigVersion.cmake"
  DESTINATION lib/cmake/Fib
)
```

cmake/FibConfig.cmake

```
include(CMakeFindDependencyMacro)
find_dependency(Boost)
include("${CMAKE_CURRENT_LIST_DIR}/FibTargets.cmake")
```

5.2.2 CPack

```
# $ cd build
# $ cmake ..
# $ make -j 2
# $ cpack -G TGZ .

cmake_minimum_required(VERSION 3.10)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED True)
project(a)

add_executable(a a.cc)
add_library(b b.cc)
target_link_libraries(a PRIVATE b)
include(GNUInstallDirs)
install(TARGETS a b
  RUNTIME DESTINATION ${CMAKE_INSTALL_BINDIR}
  ARCHIVE DESTINATION ${CMAKE_INSTALL_LIBDIR}
  LIBRARY DESTINATION ${CMAKE_INSTALL_LIBDIR}
)
install(FILES b.h DESTINATION ${CMAKE_INSTALL_INCLUDEDIR})

set(CPACK_GENERATOR "ZIP;TGZ")
SET(CPACK_DEBIAN_PACKAGE_MAINTAINER "crazyguitar")
include(CPack)
```

5.3 External Project

Table of Contents

5.3.1 Add an External Project

```
include (ExternalProject)
ExternalProject_Add(fmt
  GIT_REPOSITORY "https://github.com/fmtlib/fmt.git"
  GIT_TAG "7.1.3"
  GIT_CONFIG advice.detachedHead=false
  PREFIX "${CMAKE_BINARY_DIR}/fmt"
  CMAKE_CACHE_ARGS
    "-DFMT_INSTALL:BOOL=ON"
    "-DFMT_DOC:BOOL=OFF"
    "-DFMT_TEST:BOOL=OFF"
    "-DCMAKE_INSTALL_PREFIX:PATH=${CMAKE_BINARY_DIR}"
)
```

5.3.2 Download Only

```
include (ExternalProject)
ExternalProject_Add(fmt
  GIT_REPOSITORY "https://github.com/fmtlib/fmt.git"
  GIT_TAG "7.1.3"
  GIT_CONFIG advice.detachedHead=false
  PREFIX "${CMAKE_BINARY_DIR}/fmt"
  CONFIGURE_COMMAND ""
  BUILD_COMMAND ""
  INSTALL_COMMAND ""
)
```

5.3.3 Build via GNU Autotool

```
include (ExternalProject)
ExternalProject_Add(curl
  URL "https://github.com/curl/curl/releases/download/curl-7_74_0/curl-7.74.0.tar.gz"
  URL_MD5 "45f468aa42c4af027c4c6ddba58267f0" # md5sum curl_7.74.0.tar.gz
  BUILD_IN_SOURCE 1
  SOURCE_DIR ${CMAKE_BINARY_DIR}/curl
  CONFIGURE_COMMAND ${CMAKE_BINARY_DIR}/curl/configure --prefix=${CMAKE_BINARY_DIR}
  BUILD_COMMAND make
  INSTALL_COMMAND make install
)
```

CHAPTER

SIX

GNU DEBUGGER CHEAT SHEET

6.1 Debug with GDB

Table of Contents

6.1.1 Load an Executable

Using GDB to debug requires it recognizes a program's debug symbols. By compiling with `-g` option, GDB will understand what source code looks like after loading an executable file:

```
$ gcc -g -Wall -Werror foo.c # compile with -g option
$ gdb ./a.out  # load all symbols of a.out into GDB
```

6.1.2 Text User Interface

Text User Interface (TUI) allows developers to visualize source code and to debug like using the Integrated Development Environment (IDE) to trace problems. For a beginner, entering the TUI mode is more understandable than the command line mode. The following key bindings are the most common usages for interacting with TUI.

1. Ctrl x + a - Enter or leave the TUI mode
2. Ctrl x + o - Switch the active window
3. Ctrl x + 1 - Display one window (e.g., source code + GDB shell)
4. Ctrl x + 2 - Display two windows (e.g., source code + GDB shell + assembly)
5. Ctrl l - Refresh window

6.1.3 Basic Commands

Start/Stop a program

1. start - Run an executable file and stop at the beginning
2. run / r - Run an executable file until finish or stop at a breakpoint
3. step / s - Run a program step by step with entering a function
4. next / n - Run a program step by step without entering a function
5. continue / c - Run a program until finish or stop at a breakpoint
6. finish - Step out of the current function

Set Breakpoints

1. b line - Set a breakpoint at the given line in the current file
2. b file: line - Set a breakpoint at the given line in a given file
3. b ... if cond - Set a breakpoint when the condition is true
4. clear line - Delete a breakpoint at the given line in the current file
5. clear file: line - Delete a breakpoint at giving a line in a given file
6. info breakpoints - Display breakpoints status
7. enable breakpoints - Enable breakpoints
8. disable breakpoints - Disable breakpoints
9. watch cond - Set a watchpoint for inspecting a value

Display Stack

1. backtrace / bt - Display current stack
2. frame / f framenum - Select a frame and inspect its status
3. where - Display the current stack and the line

Print Variables

1. print / p var - Print value of the given variable
2. ptype var - Print type info of the given variable
3. info args - Print function arguments
4. info locals - Print all local variables

Reverse Run

1. record - Start recording each instruction step
2. record stop - Stop recording
3. rn - Reverse next
4. rs - Reverse step

```
int main(int argc, char *argv[]) {
    int out = 0;
    for (int i = 0; i < 10; ++i) {
        out = i * i;
```

(continues on next page)

(continued from previous page)

```
    }
    return out;
}
```

```
(gdb) b main
(gdb) r
Starting program: /home/ubuntu/a.out

Breakpoint 1, main (argc=21845, argv=0x0) at test.cc:2
2       {
(gdb) record
...
(gdb) n
(gdb) p out
$1 = 1
(gdb) rn
(gdb) rn
(gdb) p out
$2 = 0
```

Define a Function

GDB provides an original way for developers to define a customized function. The following snippet shows how to define a function to display the information of the current stack.

```
(gdb) define sf
Type commands for definition of "sf".
End with a line saying just "end".
>where
>info args
>info locals
>end
```

6.1.4 Display Memory Contents

```
int main() {
    char arr[100] = "1234567890abcdefghijklmnopqrstuvwxyz";
    return 0;
}
```

```
(gdb) " x/[format] [address expression]
(gdb) " x/[len][format] [address expression]
(gdb) x/s arr
0x7fffffffe620:     "1234567890abcdefghijklmnopqrstuvwxyz"
(gdb) x/10c arr
(gdb) x/5c arr
0x7fffffffe620:     49 '1'  50 '2'  51 '3'  52 '4'  53 '5'
(gdb) x/5b arr
0x7fffffffe620:     0x31    0x32    0x33    0x34    0x35
```

CHAPTER

SEVEN

SYSTEMD CHEAT SHEET

7.1 Systemd

Table of Contents

7.1.1 Services Management

```
# start
$ systemctl start app.service

# stop
$ systemctl stop app.service

# restart
$ systemctl restart app.service

# reload
$ systemctl reload app.service

# reload a daemon after modifying a unit
$ systemctl daemon-reload

# enable a service
$ systemctl enable app.service

# disable a service
$ systemctl disable app.service

# check status
```

(continues on next page)

(continued from previous page)

```
$ systemctl status app.service

# check is active
$ systemctl is-active app.service

# check is enabled
$ systemctl is-enabled app.service

# list all units
$ systemctl list-units

# list all timers
$ systemctl list-timers
```

7.1.2 User Services

```
$ sudo loginctl enable-linger $USER
$ mkdir -p ~/.config/systemd/user/

# allow journalctl --user -xe -f --all
$ vim /etc/systemd/journald.conf

[Journal]
Storage=persistent

# move app.service to ~/.config/systemd/user/
$ systemctl --user start app.service
```

7.1.3 Service Unit

```
# app.service
#
# $ systemctl enable app.service
# $ systemctl start app.service
# $ systemctl status app.service

[Unit]
Description=Run an application

[Service]
Type=simple
Restart=always
RestartSec=30
WorkingDirectory=/path/to/app
ExecStart=/bin/bash run.sh

[Install]
WantedBy=multi-user.target
```

7.1.4 Timer Unit

```
# job.timer
#
# $ systemctl enable job.timer
# $ systemctl start job.timer
# $ systemctl list-timers

[Unit]
Description=Run a timer

[Timer]
OnBootSec=10min
OnUnitActiveSec=1m
Unit=job.service

[Install]
WantedBy=multi-user.target
```

```
# job.service

[Unit]
Description=Run a job

[Service]
Type=oneshot
WorkingDirectory=/path/to/job/folder
ExecStart=/bin/bash run.sh

[Install]
WantedBy=multi-user.target
```

```
          Minimal form              Normalized form
  Sat,Thu,Mon..Wed,Sat..Sun → Mon..Thu,Sat,Sun *-*-* 00:00:00
      Mon,Sun 12-*-* 2,1:23 → Mon,Sun 2012-*-* 01,02:23:00
                   Wed *-1 → Wed *-*-01 00:00:00
           Wed..Wed,Wed *-1 → Wed *-*-01 00:00:00
                 Wed, 17:48 → Wed *-*-* 17:48:00
Wed..Sat,Tue 12-10-15 1:2:3 → Tue..Sat 2012-10-15 01:02:03
                *-*-7 0:0:0 → *-*-07 00:00:00
                     10-15 → *-10-15 00:00:00
        monday *-12-* 17:00 → Mon *-12-* 17:00:00
  Mon,Fri *-*-3,1,2 *:30:45 → Mon,Fri *-*-01,02,03 *:30:45
      12,14,13,12:20,10,30 → *-*-* 12,13,14:10,20,30:00
           12..14:10,20,30 → *-*-* 12..14:10,20,30:00
  mon,fri *-1/2-1,3 *:30:45 → Mon,Fri *-01/2-01,03 *:30:45
            03-05 08:05:40 → *-03-05 08:05:40
                  08:05:40 → *-*-* 08:05:40
                     05:40 → *-*-* 05:40:00
    Sat,Sun 12-05 08:05:40 → Sat,Sun *-12-05 08:05:40
          Sat,Sun 08:05:40 → Sat,Sun *-*-* 08:05:40
          2003-03-05 05:40 → 2003-03-05 05:40:00
```

(continues on next page)

(continued from previous page)

```
05:40:23.4200004/3.1700005 → *-*-* 05:40:23.420000/3.170001
             2003-02..04-05 → 2003-02..04-05 00:00:00
      2003-03-05 05:40 UTC → 2003-03-05 05:40:00 UTC
                2003-03-05 → 2003-03-05 00:00:00
                     03-05 → *-03-05 00:00:00
                    hourly → *-*-* *:00:00
                     daily → *-*-* 00:00:00
                 daily UTC → *-*-* 00:00:00 UTC
                   monthly → *-*-01 00:00:00
                    weekly → Mon *-*-* 00:00:00
   weekly Pacific/Auckland → Mon *-*-* 00:00:00 Pacific/Auckland
                    yearly → *-01-01 00:00:00
                  annually → *-01-01 00:00:00
                     *:2/3 → *-*-* *:02/3:00
```