csheeet Documentation

Release 0.1.0

crazyguitar

Dec 05, 2016

Contents

Contents:

CHAPTER 1

C Basic cheatsheet

1.1 Comma Operator

```c
#include <stdio.h>

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

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.

1.2 Old Style and New Style Function Definition

```c
#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[])
{

        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
```

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

1.3.1 Reference

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

```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[])
{
```

```
        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
```

```
#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);
         ^
```

1.4 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[])
{
    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
```

1.5 Implement closure via `static`

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

1.6 Split String

```
#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[])
{
    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
```

1.7 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)

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!
```

1.8 Duff's device

```
#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
```

1.9 `switch` goto `default` block

```
#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");
            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
```

1.10 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
#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!
```

1.11 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 {
        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
```

1.12 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);
    } 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

1.13 Implement a Task Chain

```c
#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;
    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);              \
        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
```

CHAPTER 2

C Macro cheatsheet

2.1 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
```

2.2 DEBUG switch

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

2.3 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
```

2.4 FOREACH

```c
#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");

    ret = 0;
    return ret;
}
```

output:

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

2.5 ALLOC_STRUCT

```c
#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
```

2.6 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
```

2.7 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__ }

#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
```

2.8 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
```

2.9 define `__attribute__ ((*))`

```c
#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
```

CHAPTER 3

GNU C Extensions cheatsheet

3.1 Using `__extension__` prevent `-pedantic` warning

3.1.1 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
```

3.1.2 without __extension__

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

#include <stdio.h>

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

3.2 Binary Constants

ref: Binary Constants

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

#include <stdio.h>

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
```

3.3 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));
        printf("max(%d, %d) = %d\n", a, b, max(a, b));
        return 0;
}

#endif
```

output:

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

3.4 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",
                SEARCH(arr, ARRAYSIZE(arr), target), target);
        return 0;
```

```
}

#endif
```

output:

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

3.5 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++; }
        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];
        }

        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
```

3.6 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])

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
```

3.7 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>

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
```

3.8 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));

        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[])
{
        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
```

3.9 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
```

3.10 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 */

        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
```

3.11 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;
        }
        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

3.12 Designated Initializers

ref: Initializers

3.12.1 Array initializer

```c
#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 */
        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

```c
#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");

        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
```

3.12.2 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);
        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
```

3.13 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;
};

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;
};

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
```

CHAPTER 4

C file operations cheatsheet

4.1 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;

    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
```

4.2 Using `fstat` get file size

```c
#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;
    }
    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
```

4.3 Copy all content of a file

```c
#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;
    }

    /* 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
```

4.4 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;
    }
    /* 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
```

4.5 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;
    }

    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;
}
```

4.6 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;
    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;
}
```

4.7 Check file types

```
#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
```

4.8 File tree walk

```
#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
$ ./a.out .
./tree_walk.c
./a.out
.
```

CHAPTER 5

C signal operation cheatsheet

5.1 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];
        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
```

5.2 Basic signal event handler

```c
#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)
 */

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
```

5.3 A pthread signal handler

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>

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
Get signal[3]: Quit
$ kill -TERM %1
[1]+  Terminated              ./a.out
```

5.4 Check child process alive

```
#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
```

5.5 Basic sigaction usage

```
#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
```

5.6 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
```

CHAPTER 6

C Concurrency cheatsheet

6.1 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;
}
```

6.2 Using `daemon(nochdir, noclose)`

```
#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 7

C socket cheatsheet

7.1 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;
    }

    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
```

7.2 Transform host & network endian

```c
#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);
    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
```

7.3 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[])
{
    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);
        close(c);
    }
    ret = 0
Error:
    if (s >= 0){
        close(s);
    }
    return ret;
}
```

output:

```
$ ./a.out &
[1] 63546
$ nc localhost 5566
Hello Socket
Hello Socket
```

7.4 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>
```

```c
#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;

    /* 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
```

7.5 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);
        }
    }
    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);
                    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
```

7.6 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)

#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;
    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
```

CHAPTER 8

C Makefile cheatsheet

8.1 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
foo bar
foo foo foo bar bar
```

8.2 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
```

8.3 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 ""

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 ==
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
```

8.4 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
```

8.5 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:
        @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
```

8.6 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
```

8.7 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
```

8.8 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
```

8.9 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
```

8.10 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)

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

8.11 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)
        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
|   |-- 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
```

8.12 replace current shell

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

.PHONY: all

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

output

```
$ make
Linux
```

8.13 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

$ make VAR=true
not empty
```

8.14 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
```