COMP 2355 Introduction to Systems Programming

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

Today

- sprintf
- scanf
- file IO
- fork

System Programming 101

- Check return values of all system calls
- Print appropriate error messages, especially when doing input-output operations

System Programming 102

- It is almost always ok not to check return values of `printf` and `fprintf`
- It is sometimes ok not to check `malloc` (and related functions, such as `strdup`)
- It is sometimes ok not to check functions like `close`
- The slides will often omit error checking code; this does not mean that you are ever allowed to program like that!

UNIVERSITY OF DENVER

The `printf` function family

- `int printf(const char * format, ...);`
- `int fprintf(FILE * stream, const char * format, ...);`
- `int sprintf(char * str, const char * format, ...);`
- `int snprintf(char*str, size_t size, const char*format, ...);`

Example: `sprintf` and `fprintf`

```
char buf[128];
snprintf(buf, 128, "The course has %u students.\n", 42);
fprintf(stdout, buf);
fprintf(stdout, "The course has %u students.\n", 42);
```

The `scanf` function family

- `int scanf(const char*format, ...)`
- `int fscanf(FILE*stream, const char*format, ...`
- `int sscanf(const char*str, const char*format, ...)`

Minimal Example

```
int d;
assert 1 == sscanf("42", "%d", &d);
assert d == 42;
```

Minimal Example

```
float f;
assert 1 == sscanf("4.2", "%f", &f);
assert f == 4.2;
```

Minimal Example

```
unsigned long long u;
assert 1 == sscanf("42424242", "%llu", &u);
assert u == 42424242LL;
```

Minimal Example

```
int d;
char s[2+1];
unsigned int x = 1;
assert 2 == sscanf("42 us kq",
                   "%d %2s %x",
                   &d, s, &x);
assert d == 42;
assert 0 == strcmp(s, "us");
assert x == 1;
```

Minimal Example

```
char s[100+1];
assert 1 == sscanf("Hello world!", "%100s", s);
assert 0 == strcmp(s, "Hello");
```

Minimal Example

```
char s[2+1];
assert 1 == sscanf("42 us 7f",
                   "%*d %2s %*x", s);
assert 0 == strcmp(s, "us");
```

Minimal Example

```
int a;
int b;
assert 2 == sscanf("42 99", "%1d %d", &a, &b);
assert 4 == a;
assert 2 == b;
```

GNU extension

```
char * s;
assert 1 == sscanf("HelloWorld", "%as", &s);
assert 0 == strcmp(s, "HelloWorld");
free(s);
```

More Information

- `man 3 sscanf`

File IO

File IO uses two interfaces:

- High-level interface using `FILE *` as a handle

⇒ man-pages in section 3

- Low-level interface using `int` as a handle

⇒ man-pages in section 2

Key functions: High Level API

- `FILE * fopen(const char * path, const char * mode)`
- `size_t fread(void *ptr,`
  `size_t size, size_t nmemb, FILE *stream)`
- `size_t fwrite(const void *ptr,`
  `size_t size, size_t nmemb, FILE *stream)`
- `int fflush(FILE * fp)`
- `int feof(FILE * fp)`
- `int fclose(FILE * fp)`

Standard streams

- FILE * stdin;
- FILE * stdout;
- FILE * stderr;

`fprintf(stdout, "Hello World");` $\equiv$ `printf("Hello World");`

High Level File IO and `scanf`

```
char buf[129];
unsigned long long t;
FILE * f = fopen("/proc/vmstat", "r");
while ( (2 == fscanf(f, "%128s %llu\n", buf, &t)) &&
        (0 != strcmp("nr_active", buf)) );
fclose(f);
if (0 == strcmp("nr_active", buf))
  printf("The operating system currently "
         "uses %llu pages of memory\n", t);
```

Checking and reporting errors

```
#define PROC_VMSTAT "/proc/vmstat"
FILE * f = fopen(PROC_VMSTAT, "r");
if (f == NULL) {
  fprintf(stderr, "Could not open file ‘%s’: %s\n",
          PROC_VMSTAT, strerror(errno));
  abort(); /* or other appropriate action */
}
// ...
```

Directories

- `DIR * opendir(const char * name)`
- `struct dirent * readdir(DIR * dir)`
- `int closedir(DIR * dir)`

`struct dirent`

```
struct dirent {
  ino_t            d_ino;
  char             d_name[256];
};
```

`struct dirent` may have other fields, but those are OS-specific and should not be used if it can be avoided.

Listing files

```
struct dirent * ent;
DIR * dir = opendir("/proc");
while (NULL != (ent = readdir(dir))) {
  printf("Found file ‘%s’\n", ent->d_name);
}
closedir(dir);
```

Key functions: Low Level API

- `int open(const char * path, int flags)`
- `int open(const char * path, int flags, mode_t mode)`
- `ssize_t read(int fd, void * buf, size_t count)`
- `ssize_t write(int fd, const void * buf, size_t count)`
- `int fsync(int fd)`
- `int close(int fd)`

Standard File Descriptors

- 0: stdin
- 1: stdout
- 2: stderr

`fprintf(stderr, "Hello World");` $\equiv$ `write(2, "Hello World", 11);`

Low Level API: interesting functions

- `off_t lseek(int fd, off_t offset, int whence)`
- `int dup(int oldfd)`
- `int dup2(int oldfd, int newfd)`
- `int pipe(int pipefd[2])`
- `int fcntl(int fd, int cmd, ...)`

select

- FD_ZERO(fd_set *set)
- FD_SET(int fd, fd_set *set)
- int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)

Example (1/3)

```
int pi[2];
pipe(pi);
if (fork() == 0) {
  close(pi[0]);
  close(0); close(1); close(2);
  while (1) { write(pi[1], "Hello\n", 6); sleep(5); }
} else {
  close(pi[1]);
  while (1) { merge(pi[0], 0, 1); }
}
```

Example (2/3)

```
#define MAX(a,b) ((a) > (b) ? (a) : (b))
void merge(int in1, int in2, int out) {
  fd_set rs, ws;
  FD_ZERO(&rs);      FD_ZERO(&ws);
  FD_SET(in1, &rs); FD_SET(in2, &rs);
  select(1 + MAX(in1, in2), &rs, &ws, NULL, NULL);
  if (is_set(in1, rs)) copy(in1, out);
  if (is_set(in2, rs)) copy(in2, out);
}
```

Example (3/3)

```
void copy(int in, int out) {
  size_t num;
  char buf[1024];

  num = read(in, buf, sizeof(buf));
  write(out, buf, num);
}
```

ioctl and fcntl

- int ioctl(int d, int request, ...)
- int fcntl(int fd, int cmd, ...)
- Uses:
  - locking
  - signal handling
  - non-blocking IO

Converting between the APIs

These are usually a bad idea, but you can use:

- `FILE * fdopen(int fd, const char * mode)`
- `int fileno(FILE * stream)`

Homework

- Study all of the man pages for all of the system calls discussed today.
- There are additional system calls discussed in those man pages, make sure you know them!
- For the assignment, read the man page for `readlink` and `(f)stat`.

Questions

?