Материалы к лекциям по основам системного программирования

Жмылёв Сергей Александрович

Осень 2018

УНИВЕРСИТЕТ ИТМО

Структура курса I

Системное программное обеспечение

Программный интерфейс

Структура программ

Компиляция программ

Код возврата

Роль системы

Загрузка системы

Системные вызовы

Структура курса II

Обращение к системе

Файл

Дескриптор файла

Потоки ввода-вывода процесса

Стандартные потоки ввода-вывода

open(2)

Основные режимы доступа

lseek(2)

Структура курса III

- read(2)
- write(2)
- close(2)
- dup(2) и dup2(2)
- stat(2)
- Ошибки в системных вызовах
- Стандартизация ошибок
- Пример ошибки

Структура курса IV

Заголовочные файлы

Пример чтения/записи

Makefile

Утилита make

Ввод-вывод со смещением

Полезные ссылки

http://src.illumos.org/

https://github.com/mit-pdos/xv6-public

https://se.ifmo.ru/~korg/

https://vk.com/korglings

Литература:

1. Uresh Vahalia. UNIX Internals
2. A. S. Tanenbaum, A. S. Woodhull. Operating Systems: Design and Implementation

Системное программное обеспечение

- Языки СПО
- Системные вызовы
- Ввод-вывод
- Потоки и процессы

Программный интерфейс

Любая программа принимает аргументы и переменные окружения

Код возврата – число, отображающее корректность завершения

Структура программ

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

    return 0;
}
```

Структура программ

```
#include <stdlib.h>

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

    return EXIT_SUCCES;
}
```

Компиляция программ

```
# gcc -c program.c
# gcc -o program program.o

# gcc -o program program.c

# cc -o program program.c
```

Код возврата

```
# rm -f /etc/passwd 2<&-
# echo $?
1
# echo Hello, world!
Hello, world!
# echo $?
0
```

Популярная ошибка

Использование void main() - недопустимо!

```
# cat void.c
void main(void) {}
# ./void
# echo $?
16
```

Роль системы

- Многозадачность;
- Виртуализация памяти;
- Управление устройствами;
- Обработка прерываний;
- Расширение набора операций, доступных программам.

Загрузка системы

- Reset vector: UEFI, BIOS, . . .
- I/O, *PIC(IRQ), VGA
- POST + PCI BIOS
- Выбор загрузочного устройства
- Загрузчик stage0 (boot sector)
- Загрузчик stage1
- Ядро ОС

Системные вызовы

- Обращение к функции ядра системы
- Использование аппаратного обеспечения через единый API
- Имеют интерфейс libc
- Имеют привилегии ядра операционной системы

Обращение к системе

```
/* завершение программы с кодом 2 */
_exit(2);
```

```
.globl _start
_start:
pushq $2
movq $1, %rax
int $0x80
```

```
# cc -m64 -Wall -Wextra -Wno-comment \
   -nostdlib -o main main.S
```

https://pastebin.com/knTdpZRe

Что такое файл?

Everything is a file!
Кроме потоков и ядра

Дескриптор файла

`http://src.illumos.org/source/xref/illumos-gate/usr/src/uts/common/syscall/open.c#54`

`http://src.illumos.org/source/xref/illumos-gate/usr/src/uts/common/fs/vnode.c#940`

- Номер файлового дескриптора – целое неотрицательное число, абстрагирующее процессы от файлов, с которыми они работают.

Потоки ввода-вывода процесса

Номер	Файл	Флаги
0	/dev/tty	O_RDWR | O_LARGEFILE
1	/dev/tty	O_RDWR | O_LARGEFILE
2	/dev/tty	O_RDWR | O_LARGEFILE
3	/etc/passwd	O_RDONLY
4	/dev/mtdblock3	O_RDWR
...	...	...
255	...	...

Стандартные потоки ввода-вывода

```
# grep FILENO /usr/include/unistd.h
#define STDIN_FILENO 0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2
```

open(2)

```
int open(
    const char *path,  /* путь к файлу */
    int oflag,         /* режим доступа */
    /* mode_t mode */ /* права доступа */
);
```

Функция возвращает номер дескриптора или код ошибки

Основные режимы доступа

O_RDONLY – только для чтения

O_WRONLY – только для записи

O_RDWR – для записи/чтения

O_CREAT – создать, если не существует

O_APPEND – запись с конца

O_TRUNC – запись с начала

O_LARGEFILE – длинная позиция в файле

lseek(2)

```
off_t lseek(
   int fildes, /* номер открытого файла */
   off_t offset,    /* смещение позиции */
   int whence                  /* действие */
);
```

Функция возвращает полученное смещение в байтах или код ошибки

read(2)

```
ssize_t read(
   int fildes, /* номер открытого файла */
   void *buf,              /* буфер чтения */
   size_t nbyte        /* количество байт */
);
```

Функция возвращает количество прочитанных байт или код ошибки

write(2)

```
ssize_t write(
   int fildes, /* номер дескриптора */
   const void *buf, /* буфер записи */
   size_t nbyte  /* количество байт */
);
```

Функция возвращает количество записанных байт или код ошибки

close(2)

```
int close(
   int fildes, /* номер дескриптора */
);
```

Функция возвращает 0 или код ошибки

dup(2) и dup2(2)

```
int dup(
    int fildes /* номер открытого файла */
);
int dup2(int fildes, int fildes2);
```

Функция возвращает номер нового дескриптора или код ошибки

stat(2)

```
int stat(
   const char *restrict path,
      /* путь к файлу*/
   struct stat *restrict buf
      /* результат */
);
```

Функция возвращает номер 0 или код ошибки

Ошибки в системных вызовах

Код возврата системного вызова:

- меньше 0 – ошибка в ходе выполнения вызова
- равен 0 – успешное выполнение
- больше 0 – результат успешного выполнения

Стандартизация ошибок

- Унификация ошибочных кодов
- Переменная `errno`
- Функция `perror(3)`
- Функция `strerror(3)`

Пример ошибки

```
if (read(7, buf, 1) < 0) {
    fprintf(stderr, "%d␣", errno);
    perror("read");
    _exit(1);
}

/* 9 read: Bad file number */
```

Заголовочные файлы

- unistd.h – объявления UNIX
- stdio.h – стандартный ввод/вывод
- fcntl.h – операции с файлами
- sys/types.h – системные типы
- sys/stat.h – системные статусы

Пример чтения/записи

```
#include <unistd.h>
int main(int argc, char *argv[]) {
    int bytes;
    char buf[256];
    while((bytes = read(STDIN_FILENO, buf,
    ↪  sizeof(buf))) > 0) {
        if (write(STDERR_FILENO, buf, bytes)
    ↪    < 0) {
            return 1;
        }
    }

    return bytes;
}
```

Makefile

```
PROJS=main
CC=gcc
CFLAGS=-m64

all: $(PROJS)
	@echo Done!

$(PROJS):
	$(CC) $(CFLAGS) -o $@ $(@:=.c)
```

Утилита make

```
# make
gcc -m64 -o main main.c
Done!
# ./main
Hello, world!
```

Ввод-вывод со смещением

```
ssize_t pread(int fildes, void *buf,
   ↪  size_t nbyte, off_t offset);

ssize_t pwrite(int fildes, const void *buf
   ↪  , size_t nbyte, off_t offset);
```

Функции возвращают количество байт или код ошибки

Благодарности

- Афанасьев Дмитрий Борисович
- Горская Александра Андреевна
- Ховалкина Ксения Николаевна
- Киреев Валерий Юрьевич
- и многие другие...

Спасибо за внимание

```
# perl '-es!!),-#(-.?{<>-8#=..#<-*}>;*7-86)!;y!#() -?{}!\x20/'-v;<!;s++$_+ee'
```