Семинар 3

> Часть 1: open/pread/readv/read/write/seek
>
> Часть 2: opendir/mkdir/time/stat/stat64/fuse

Часть 0: syscalls

> open/pread/opendir/... это интересно, но про это потом

syscalls

На лекции уже пару (адын) раз говорили про сисколы, но давайте вспомним

syscalls

На саааааамой первой лекции было что-то про операционные системы

Помните что-то про них?

syscalls

На сааааааамой первой лекции было что-то про операционные системы

Помните что-то про них?

Короче да, это штука для распределения ресурсов системы

- процессорного времени
- памяти
- "интернет" ресурс
- доступ к диску
- доступ к звуковой карте
- ...

syscalls

> ... распределение ресурсов системы

Тут может возникнуть один справедливый вопрос - а как получить кусочек ресурса?)

syscalls

> ... распределение ресурсов системы

Тут может возникнуть один справедливый вопрос - а как получить кусочек ресурса?)

На этот вопрос отвечает стандарт *POSIX*

syscalls

Если долго гуглить, найдете что-то такое

Name	Entry point	Implementation
read	sys_read	fs/read_write.c
write	sys_write	fs/read_write.c
open	sys_open	fs/open.c
close	sys_close	fs/open.c
stat	sys_newstat	fs/stat.c
...	...	...

syscalls

Умный термин - *POSIX* system calls

Name	Entry point	Implementation
read	sys_read	fs/read_write.c
write	sys_write	fs/read_write.c
open	sys_open	fs/open.c
close	sys_close	fs/open.c
stat	sys_newstat	fs/stat.c
...	...	...

syscalls

Идея была в том, чтоб (ну хоть как-то) унифицировать взаимодействие программы с ОС

- *POSIX* во многом списан с UNIX
- Unix (и Unix-like) частично совместимы с *POSIX*
- В частности, Linux частично совместим с *POSIX*

syscalls

Ну и syscalls (aka системные вызовы) являются интерфейсом для взаимодействия с операционной системы.

syscalls

Ну и syscalls (aka системные вызовы) являются интерфейсом для взаимодействия с операционной системы.

(на вопрос *в куда* операционная система выделяет ресурсы поговорим потом)

syscalls

Ну и т.к. этот интерфейс единый, он имеет не так много общего с реальностью (да и не должен)

т.к. *POSIX* является стандартом именно интерфейса (все детали реализации - os depends)

Работа с файлами в Unix

Работа с файлами в Unix

Допустим мы хотим взаимодействовать с каким-то 'железом'

- повзаимодействовать с диском
- что-то воспроизвести на звуковой карте
- клавамышь (да и любой HID)
- ...

как мы это можем сделать?

Работа с файлами в Unix

Для взаимодействия с устройствами подходов в целом не то чтоб много

- можно писать что-то по каким-то адресам
- или можем придумать какую-то штуку (aka API) для взаимодействия с каждым типом устройств
- можем максимально интегрироваться в систему и быть user-friendly и сказать заветное

  > **всё есть файл!**

Работа с файлами в Unix

всё есть файл

- вот прям всё
- файлы тоже файлы
- и директория
- и звуковая карта
- и сетевое устройство
- любое устройство, которое вы подключили подхватит какой-нибудь kernel module и девайс (для непосредственного использования) будет доступен в виде файла

Работа с файлами в Unix

Если вдруг хотите поиграть с kernel modules, то

- **заведите виртуалку / возьмите rpi / возьмите не основной компьютер**
- выделите день-два
- что-то базовое вы напишите

Работа с файлами в Unix

В *POSIX* работа с файлами реализована через абстракцию **файловый дескриптор**

если кратко:

- это чиселка
- эту чиселку вам выдаёт ОС в момент открытия файла (минимальный свободный номер)
- эта чиселка связана с файлом
- эта 'связь' существует только внутри конкретного процесса (*)

НЛО - man pages

> сейчас маленькое отступление, чтоб упростить вам жизнь

НЛО - man pages

В Unix(-like) стандартный способ установки приложений - использовать менеджеры пакетов

- `apt` - Debian and Debian' (Ubuntu, LinuxMint, ...)
- `pacman` - Arch and Arch' (Manjaro, BlackArch, ...)
- `rpm / dnf` - Red Hat and Red Hat' (Fedora, CentOS)
- `pkg` - Alpine ...

НЛО - man pages

И в 'пакетах' часто (хорошая практика) поставляется документация на бинарник / библиотеку

НЛО - man pages

И в 'пакетах' часто (хорошая практика) поставляется документация (`man pages`) на бинарник / библиотеку

Доступ к этой документации осуществляется через `man`

НЛО - man pages

```
$ man ls

LS(1)                  User Commands                  LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current
       directory  by  default).  Sort entries alpha-
       betically if none of -cftuvSUX nor --sort
       ...
```

НЛО - man pages

```
$ man man

MAN(1)           Manual pager utils           MAN(1)

NAME
       man  -  an  interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       ...
DESCRIPTION
       man is the system's manual pager.  Each  page
       argument given to man is normally the name of
       a  program,  utility or function
       ...
```

НЛО - man pages

А ещё в man есть секции

Например есть

- `exit` - syscall
- `exit` - stdlib function
- `exit` - bash command

Как открыть нужное?

Мы не знаем что делать $\rightarrow$ идём курить мануал!

НЛО - man pages

А ещё в man есть секции

Например есть

- `exit` - syscall
- `exit` - stdlib function
- `exit` - bash command

Как открыть нужное?

Мы не знаем что делать $\rightarrow$ идём курить мануал!
Если очень боитесь терминала, можете гуглить `man open`

НЛО - man pages

```
$ man man

MAN(1)           Manual pager utils           MAN(1)

NAME
       man  -  an  interface to the system reference manuals
       ...
DESCRIPTION
       ...
       The table below shows the section  numbers  of  the  manual
       followed by the types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous  (including  macro  packages  and conven-
           tions), e.g. man(7), groff(7), man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]
       ...
```

НЛО - man pages

```
$ man man

MAN(1)           Manual pager utils           MAN(1)

NAME
       man  -  an  interface to the system reference manuals
       ...
DESCRIPTION
       ...
       The table below shows the section  numbers  of  the  manual
       followed by the types of pages they contain.

  ---> 1   Executable programs or shell commands
  ---> 2   System calls (functions provided by the kernel)
  ---> 3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous  (including  macro  packages  and conven-
           tions), e.g. man(7), groff(7), man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]
       ...
```

НЛО - man pages

Ну и так можем открыть

```
$ man 1 exit
$ man 2 exit
$ man 3 exit
```

1,2,3,n - номера секций

НЛО - man pages

Ну и так можем открыть

```
$ man 1 exit
$ man 2 exit
$ man 3 exit
```

Чтоб третья команда работала (допустим у вас Ubuntu / LinuxMint / Debian') нужно дополнительно догрузить страницы

```
apt-get install manpages-dev
apt-get install manpages-posix-dev
```

НЛО - man pages

Ну и так можем открыть

```
$ man 1 exit
$ man 2 exit
$ man 3 exit
```

Чтоб третья команда работала (допустим у вас Ubuntu / LinuxMint / Debian') нужно дополнительно догрузить страницы

```
apt-get install manpages-dev
apt-get install manpages-posix-dev
```

(если у вас не Ubuntu по своей воле - вы продвинутый пользователь, нагуглите сами)

Работа с файлами в Unix

список ниже покроет 99% задач (и кажется все домашки)

- open
- pread
- readv
- read
- write
- lseek

Работа с файлами в Unix

Ну погнали)

```
$ man 2 open
```

Работа с файлами в Unix

Ну погнали)

```
$ man 2 open
```

шутка

(я не буду показывать man каждой команды)

Работа с файлами в Unix

```c
int main() {
    // открываем `man 2 open` и смотрим, что нам нужно
    int fd = open("abc.txt", O_RDONLY);

    // открываем `man 2 open` и читаем, как это работает
    ssize_t len = read(fd, buff, sizeof(buff));

    // что? правильно! `man 2 lseek`
    lseek(fd, ...);

    // снова `man 2 close`
    close(fd);
}
```

Не забывайте обрабатывать ошибки *каждой* функции