ОНЛАЙН-ОБРАЗОВАНИЕ

Знакомство

С Linux и не только

Александр Румянцев

Краткая история

Первое упоминание - 03.07.91
https://groups.google.com/d/topic/comp.os.minix/T9SjMGTSpXk/discussion

Первый релиз - 25.08.91 - считается днём рождения
https://groups.google.com/d/topic/comp.os.minix/dlNtH7RRrGA/discussion

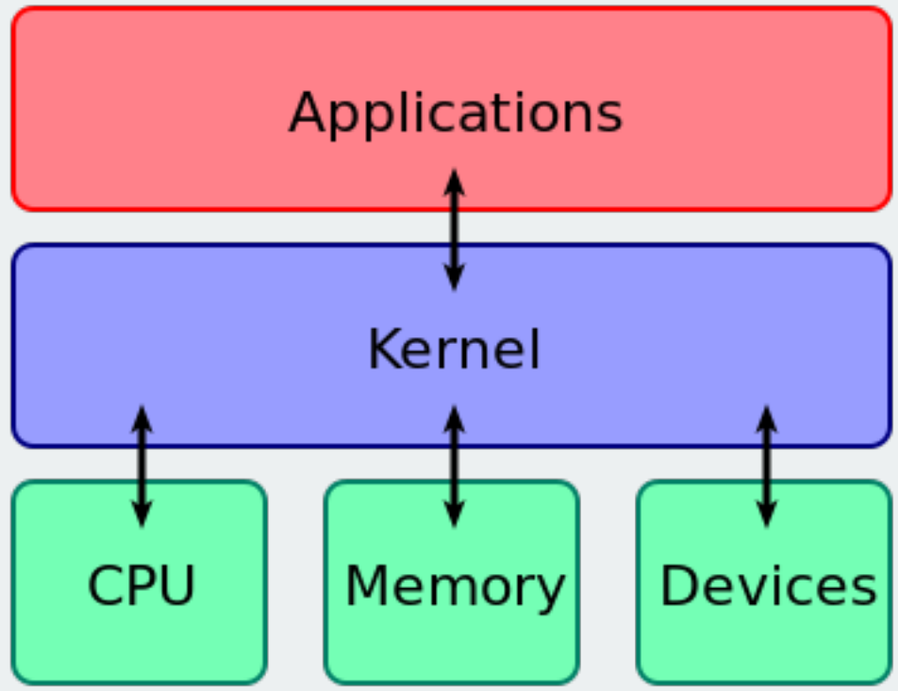
GNU - набор утилит (userland), Linux - ядро, вместе - GNU/Linux

Первый дистрибутив - Softlanding Linux System (1992) → Slackware (1993)

Значимые дистрибутивы
- **Debian (1993) → DEB based distros**
- **Red Hat (1994) → RPM based distros**
- **Yellow Dog Linux (1999) → YUM (yellowdog update manager)**

Функции ядра

- **Распределение ресурсов (scheduling) между задачами**
- **Унифицированный доступ к железу (API)**



Многозадачность

Что такое многозадачность?

Александр Румянцев

Многозадачность

Что такое многозадачность?

Свойство операционной системы обеспечивать возможность параллельной или псевдопараллельной обработки нескольких задач

Первая многозадачная операционная система - Multics (Multiplexed Information and Computing Service), прародитель UNIX (UNified Information Computer System) - 1969 год

Александр Румянцев

Виды многозадачности

1. Кооперативная
2. Вытесняющая

Подвиды многозадачности

- Системы реального времени
- Пользовательская кооперативная

Linux - вытесняющая многозадачность

Кооперативная многозадачность

Переключение контекста инициирует приложение
+ отсутствие необходимости синхронизации
– ненадёжность

Вытесняющая многозадачность

Переключение контекста инициирует ОС по прерыванию таймера
+ надёжность
– необходимость в неэффективных механизмах синхронизации

Linux - вытесняющая многозадачность

Александр Румянцев

Виды ядер

Монолитное (Linux)
Микроядро (GNU Mach, QNX)
Гибрид (Windows)
Почти монолит (Mac, Xen)

Процессор. Регистры

Регистры общего назначения
AX, BX, CX, DX

Регистры селекторов
CS (Code Segment)
SS (Stack Segment)

Указатели состоят из селектора и смещения
CS:IP - указывает на адрес текущей исполняемой инструкции
SS:SP - указывает на вершину стека

Александр Румянцев

Процессор. Память

Все современные архитектуры не адресуют память напрямую. Вместо этого имеется минимум два уровня абстракции

- **таблицы дескрипторов GDT и LDT в процессоре, преобразующие указатели в логический адрес**
- **устройство MMU, преобразующий логический адрес в физический адрес**

Каждый процесс имеет свою LDT, и каждый процесс не имеет доступа к памяти другого процесса.

Процессор. Уровни выполнения (Ring)

В каждой записи LDT и GDT хранятся уровни привелегий от 0 до 3.

Если уровень привелегий кода <= уровень привелегий данных, то доступ разрешен, иначе - GPF он же Page Fault

Т.О. уровень 0 - наивысший.

Кроме того, некоторые инструкции могут работать только с уровнем привелегий 0

Процессор. Стек

Область памяти, общение с которой происходит по принципу LIFO
В ней хранится состояние регистров, параметры функций, локальные переменные функций

Вызов функции

```
MOV AX, 0x1
PUSH ax
CALL function
```

В момент CALL процессор в стек помещает еще пару CS:IP

Функция

```
POP AX
INC AX
PUSH AX
RET
```

В момент RET процессор восстанавливает из стека предыдущее значение CS:IP
За тем, что должно быть в стеке на момент возврата следит компилятор. Или не следит.

Процессор. Прерывания

Прерывания - это функции, вызывающиеся по какому-то событию
Адреса этих функций хранятся IDT - Interrupt Description Table
Номер прерывания - номер в таблице IDT

Прерывания бывают:

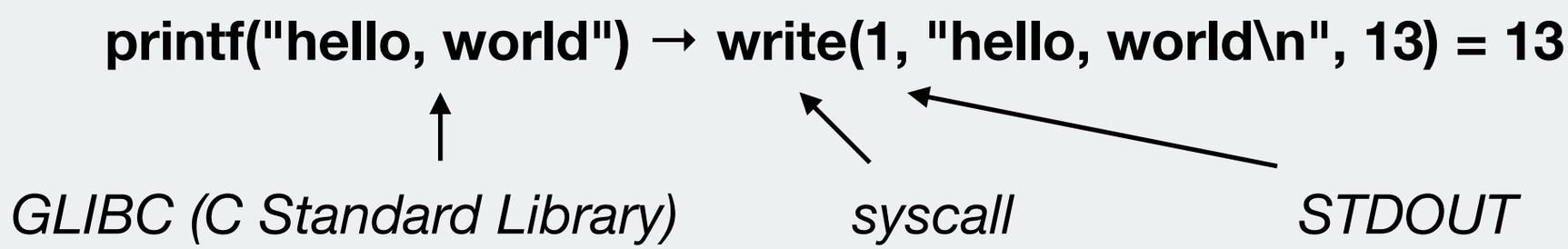
- **Аппаратные (если подано напряжение на ногу процессора)**
- **Исключения (если произошла исключительная ситуация в коде)**
- **Программные (вызываются в коде инструкцией INT)**

Процессор обрабатывает прерывание как и вызов функции, за исключением того, что в стек кладётся еще и регистр флагов

Функции ядра (API) Linux - syscall

Библиотечные функции, исполняемые в контексте текущей задачи, вызываемые через програмные прерывания (int 80) или специальную инструкцию процессора (syscall), выполняющиеся с повышенными привилегиями (ring 0)

При вызове syscall сохраняется состояние задачи (некоторые регистры и флаги) в стек, т.е. вызов syscall - дорогое удовольствие. Как пример - gettimeofday().



Загрузка ядра и процессы ядра

В процессе выполнения кода ядра (загрузка) проходят следующие этапы:

- **Инициализация механизмов управления реурсами CPU (установка шедулера в обработчик прерывания таймера)**
- **Инициализация системы управления памятью (установка обработчиков исключений, инициализация таблиц аллокаций)**
- **Инициализация системы управления процессами (инициализация очереди сигналов)**
- **Инициализация драйверов устройств (настройка устройств и установка обработчиков прерываний устройств)**
- **Запуск ядерных процессов**
- **Запуск первого процесса**

Переключение контекста на x86

Контекст - текущее состояние регистров процессора (общего назначения, указатель на инструкцию и указатель на стек, флаги).

Переключение на другую задачу происходит через шедулер, вызываемый аппаратным прерыванием таймера, с помощью подмены указателя на вершину стека, после чего из стека загружаются сохранённые регистры и делается возврат из прерывания уже в другую задачу.

По сути шедулер оперирует таблицей указателей на вершину стека

В некоторых процессорных архитектурах переключение контекста облегчается через "регистровый файл" - сотня-другая регистров общего назначения, из которых приложению видны только регистры из "регистрового окна" с некоторым смещением от начала "файла". Переключение контекста осуществляется сдвигом окна.

Сигналы

Сигнал - указание шедулеру, что процесс надо поставить на исполнение не с последней инструкции, на которой он был прерван, а с сохраненной в таблице обработчиков сигналов функции

Обработчики сигналов устанавливаются программой.
Сигнал с номером 9 не передаётся процессам, шедулер просто инициирует завершение процесса.
Сигнал номер 9 игнорируется для процесса с номером 1

Сигналы инициируются ядром (в случае исключительных ситуаций) или другими процессами

HZ

```
# zgrep CONFIG_HZ= /proc/config.gz
CONFIG_HZ=1000
```

```
# zgrep CONFIG_HZ= /boot/config-*
CONFIG_HZ=1000
```

LA

Количество задач, стоящих в очереди на исполнение

Трассировка

Утилиты для трассировки

```
strace - трассировка системных вызовов
ltrace - трассировка библиотечных вызовов
```

Посмотрим, что это на примере "hello, world"

Александр Румянцев

Linux версии

Система нумерации до 3.0

2.6.32

минор четный - стабильный релиз
минор нечетный - ветка разработки

Система нумерации после 3.0

Как попало

mainline
stable
longterm (LTS)
linux-next

У разных дистрибутивов возможно разное понимание LTS

https://www.kernel.org/

Ядра для RHEL/CentOS

Штатные

```
yum update kernel
yum remove kernel
```

ELRepo - "ванильные ядра" (http://elrepo.org/tiki/tiki-index.php)

```
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
```

`kernel-lt` - longterm
`kernel-ml` - mainline

OUEK - Oracle Unbreakable Enterprise Kernel
curl -o /etc/yum.repos.d/ouek.repo http://yum.oracle.com/public-yum-ol7.repo

Модули и параметры ядра

Унификация образов ОС, модули подгружаются по мере необходимости
Уменьшение размера образа ядра

Утилиты для работы с модулями:

```
lsmod
modprobe
insmod
rmmod
```

Конфигурация модулей:

```
/sys/modules
/etc/modprobe.d
/etc/modules-load.d
```

Александр Румянцев

Своё ядро

```
cp /boot/config* .config &&
make oldconfig &&
make &&
make install &&
make modules_install
```


Домашка

1. Результаты складываем в github
2. Ссылку присылаем в "чат с преподавателем"

Александр Румянцев

Спасибо за внимание!

Вопросы?