Архитектура ЭВМ

Лекция 6. Системные вызовы

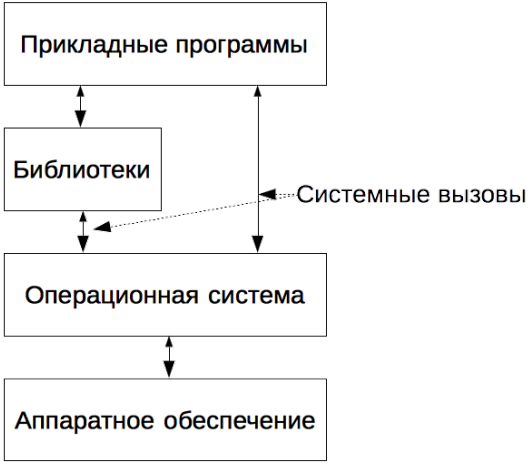
к.ф.-м.н. Филонов Павел Владимирович
filonovpv@gmail.com

Московский Государственный Технический Университет
Гражданской Авиации

Содержание

Взаимодействие с ОС

Системный вызов (англ. system call) в программировании и вычислительной технике — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.



Прерывания процессора

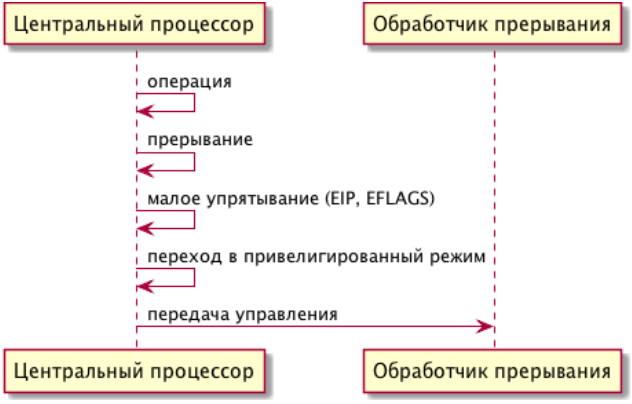
- внешние (аппаратные)
- внутренние (ловушки)
- программные (системные вызовы)

Внешние прерывания

Данный тип прерывания используется аппаратным обеспечением (таймер, контроллер внешнего носителя, сетевая карты и т.д.) для сигнализации ЦП об наступлении какого-либо события (срабатывание таймера, завершение операции записи, приход сетевого пакета и т.д.)

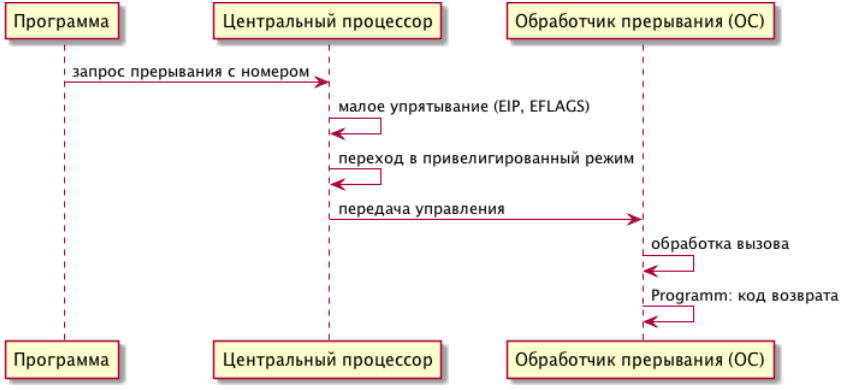
Внутренние прерывания

Данный тип прерываний используется ЦП для передачи управления обработчикам его собственных внутренних событий (деление на ноль, ошибка сегментации и т.д.).



Программные прерывания

Используются для передачи управления из прикладной программы в ОС (системные вызовы)



Соглашение об системных вызовах в ОС Linux

- номер системного вызова передается через `eax`
- параметры передаются через следующие регистры:
  - `ebx, ecx, edx, esi, edi`
- системный вызов осуществляется через прерывание по номеру 0x80
  - `int 0x80`
- результат выполнения возвращается через `eax`
  - результат в диапазоне от `0xfffff000` до `0xffffffff` сигнализирует об ошибке

Пример: системный вызов exit

Используется для завершения работы программы. В качестве параметра принимает целое число — код завершения. Любое число отличное от нуля означает неудачное завершение программы.

```
1 ; exit(0)
2 mov eax, 1 ; 1 - номер системного вызова exit
3 mov ebx, 0 ; 0 - значение параметра (нормальное завершение программы)
4 int 0x80   ; обращение к прерыванию номер 0x80
```

Пример: системный вызов write

Используется для записи данных. В качестве параметров принимает:

1. номер файлового дескриптора для вывода;
2. адрес участки памяти, который необходимо записать;
3. число байт, которое необходимо записать.

```
1  section .data
2      msg db "Hello, world!", 10
3      len equ $ - msg
4  section .text
5  _start:
6      ; write(1, msg, len);
7      mov eax, 4    ; 4 - номер системного вызова write
8      mov ebx, 1    ; 1 - номер файлового дескриптора,
9                    ; который связан со стандартным потоком вывода
10     mov ecx, msg  ; адрес первого байта строки
11     mov edx, len  ; длина строки
12     int 0x80      ; обращение к прерыванию номер 0x80
```

Пример: системный вызов read

Используется для чтения данных. В качестве параметров принимает:

1. номер файлового дескриптора для записи
2. адрес участка памяти, куда необходимо записать данные;
3. максимальное число байт, которое можно прочитать.

Возвращает код ошибки или число успешно прочитанных байт. Если возвращается 0, то это означает, что достигнут «конце файла».

```
1  section .bss
2      buf resb 128
3  section .text
4  _start:
5      ; read(0, buffer, 128)
6      mov eax, 3   ; 3 - номер СВ read
7      mov ebx, 0   ; 0 - номер файлового дескриптора,
8                   ; который связан со стандартным потоком ввода
9      mov ecx, buf ; в какой участок памяти записать данные
10     mov edx, 128 ; максимальное число байт, которое можно прочитать
11     int 0x80
```

Соглашение об системных вызовах в ОС FreeBSD

- номер системного вызова записывается в регистре `eax`;
- параметры передаются через стек;
- вызов осуществляется в специальной функции `kernel`.

```
1  kernel:
2      int 0x80
3      ret
4
5  write:
6      push dword len
7      push dword msg
8      push dword 1
9      mov eax, 5
10     call kernel
11     add esp, 12
```

Использование команды sysenter

Является более быстрой версией `int 0x80`

- номер системного вызова сохраняется в `eax`;
- параметры сохраняются в регистрах `ebx, ecx, edx, esi, edi`
- на стеке сохраняется адрес возврата;
- на стеке сохраняются регистры `ecx, edx, ebp`;
- адрес верхушки стека сохраняется в `ebp`;
- выполняется машинная команда `sysenter`;
- возвращаемое значение сохраняется в `eax`.

Пример использования sysenter

```
1  _start:
2      ; write(1, msg, len)
3      mov eax, 4
4      mov ebx, 1
5      mov ecx, msg
6      mov edx, len
7      push dword write_ret
8      push ecx
9      push edx
10     push ebp
11     mov ebp, esp
12     sysenter
13
14 write_ret:
15     ; ...
```

Использование команды syscall

Используется в наборе машинных команд x86_64

- номер системного вызова сохраняется в rax;
- номера системных вызовов отличаются от x86;
- параметры сохраняются в регистрах rdi, rsi, rdx, r10, r8, r9;
- выполняется машинная команда syscall;
- возвращаемое значение сохраняется в rax.

```
1  _start:
2      ; write(1, msg, len)
3      mov rax, 1 ; 1 - номер СВ write для системы команд x86_64
4      mov rdi, 1
5      mov rsi, msg
6      mov rdx, len
7      syscall
```

Резюме

- Определение и типы прерываний:
  - внешние;
  - внутренние;
  - программные.
- Программы взаимодействуют с ОС посредством программных прерываний (системных вызовов)
- Примеры системных вызовов:
  - `exit`;
  - `write`;
  - `read`.
- Соглашения об системных вызовах:
  - `int 0x80` для Linux;
  - `int 0x80` для FreeBSD;
  - `sysenter` для Linux;
  - `syscall` для Linux x86_64.