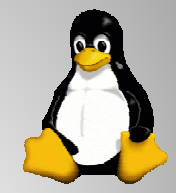


Chamadas de Sistema

Onofre Trindade Jr

Chamadas de Sistema

- Interface entre um processo do usuário e dispositivos de hardware
  - CPU, memory, disks etc
- Facilidade de programação
  - O kernel se responsabiliza pelas tarefas de baixo nível
- Maior segurança
  - Baseada no kernel através das chamadas de sistema
- Portabilidade:
  - A interface (API) é mantida mesmo que a implementação seja trocada (p.ex. Kernel 2.4 para 2.6)

API POSIX

- API = Application Programming Interface
  - Definição da interface das funções para obtenção de serviços
  - As chamadas do sistema são feitas através de interrupções de software (instrução **int 0x80**, para processadores Intel e kernels 2.4 e anteriores)
- A biblioteca C padrão (**libc**) contém rotinas de encapsulamento (wrapper) para fazer as chamadas de sistema
  - e.x., **malloc** e **free** são rotinas libc que utilizam a chamada de sistema **brk**
- POSIX => conjunto padrão de APIs.

Chamadas de Sistema no Linux

- Chamadas executando **`int $0x80`**
  - Kernel 2.4 e anteriores e processadores Intel
  - Processo que chama passa o número da **`syscall no registrador eax`**
- O handler da Syscall é responsável por:
  - Salvar registradores na pilha do kernel
  - Chamar a rotina de serviço da syscall
  - Na saída, executa **`ret_from_sys_call()`**.

- 1 General commands
- 2 System calls
- 3 C library functions
- 4 Special files (usually devices,/dev) and drivers
- 5 File formats and conventions
- 6 Games and screensavers
- 7 Miscellanea
- 8 System administration commands and daemons

Páginas de Manual no Linux

- Tabela de syscalls:
  - Associa cada número de syscall com o endereço da rotina de atendimento correspondente
  - Armazenada em **`sys_call_table`**, contendo até **`NR_syscall`** entradas (usualmente 256)

Chamadas de Sistema no Linux

A Função system_call ()

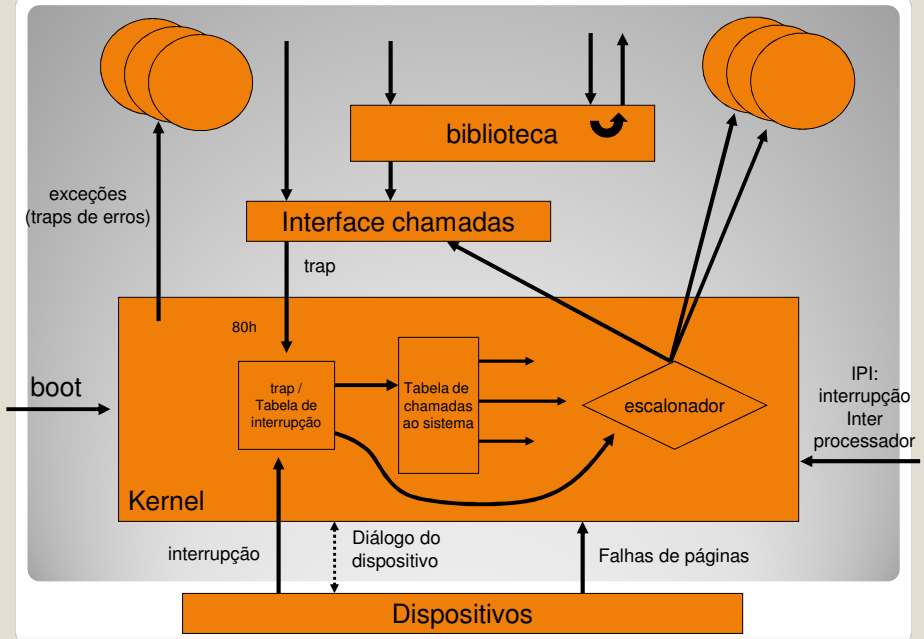
- Salva o número da syscall e registradores da CPU na pilha
- Verifica se a chamada é válida
- Chama a rotina de atendimento associada com o número contido no registrador eax
  - `call *sys_call_table(0, %eax, 4)`
- O código de retorno da syscall é armazenado em **`eax`**.

Parâmetros

- Na arquitetura Intel 80x86 de 32-bit:
  - 6 registradores são utilizados para a passagem de parâmetros
    - **`eax`** (número da syscall).
    - **`ebx`**, **`ecx`**, **`edx`**, **`esi`**, **`edi`** , armazenam parâmetros

Rotinas de Encapsulamento

- As macros **`_syscall0 … _syscall6`** definem rotinas de encapsulamento para chamadas com até 6 parâmetros
- e.x.
  - **`_syscall3(int,write,int,fd,const char, *buf,unsigned int, count)`**



```
.data                                   # section declaration

msg:
        .string "Hello, world!\n"       # our dear string
        len = . - msg                   # length of our dear string

.text                                   # section declaration

                        # we must export the entry point to the ELF linker or
    .global _start      # loader. They conventionally recognize _start as their
                        # entry point. Use ld -e foo to override the default.

_start:

# write our string to stdout

        movl    $len,%edx       # third argument: message length
        movl    $msg,%ecx       # second argument: pointer to message to write
        movl    $1,%ebx         # first argument: file handle (stdout)
        movl    $4,%eax         # system call number (sys_write)
        int     $0x80           # call kernel

# and exit

        movl    $0,%ebx         # first argument: exit code
        movl    $1,%eax         # system call number (sys_exit)
        int     $0x80           # call kernel
```

Exemplo: Hello World em Assembly

- Arquivos principais:
  - arch/x86/kernel/entry_32.S
    - Entrada das chamadas de sistema e rotinas de gerenciamento de baixo nível
  - include/asm-x86/unistd_32.h
    - Números das chamadas e macros
  - kernel/sys.c
    - Rotinas de atendimento

Arquivos Relacionados com as Syscalls