

## Chamada de sistema

### Interface com o Sistema Operacional

- Implementada como uma *soft interrupt*
    - Instrução: `int NUM_INT`
  - Quando uma interrupção ocorre:
    - o processador para o processo em execução
    - entra no modo supervisor
    - procura na tabela de interrupção pelo manipulador adequado (baseado no número da interrupção)
  - Chamando o sistema operacional usando interrupções, garante que o código do SO será executado em modo privilegiado
  - Utiliza os registradores para passagem de parâmetros
- [http://docs.cs.up.ac.za/programming/asm/derick\\_tut/syscalls.html](http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html)

# Exemplos

%eax	Name	Source	%ebx	%ecx	%edx	%esi	%edi
1	sys_exit	kernel/exit.c	int				
2	sys_fork	arch/i386/kernel/process.c	struct pt_regs				
3	sys_read	fs/read_write.c	unsigned int	char *	size_t		
4	sys_write	fs/read_write.c	unsigned int	const char *	size_t		
5	sys_open	fs/open.c	const char *	int	int		
6	sys_close	fs/open.c	unsigned int				

# Exemplo write

```
char test[] = "Hello world\n ";

int main()
{
    asm(" mov $4, %eax");
    asm(" mov $1, %ebx");
    asm(" mov $test, %ecx");
    asm(" mov $11, %edx");
    asm(" int $0x80");

    return 0;
}
```

## Tabela de chamada de sistema

- Tabela
  - arch/i386/kernel/syscall\_table.S
- Prototipo da função
  - include/linux/syscalls.h
- Tipos de dados
  - include/linux/types.h
- Implementação
  - kernel/sys.c
  
- `copy_to_user(destino, origem, n_bytes)`

## Criando uma chamada de sistema

1. Criando uma nova chamada
2. Adicione a chamada no arquivo `arch/i386/kernel/syscall_table.S`
  - 2.1 Adicione o protótipo da chamada de sistema no arquivo `syscalls.h`
3. Implemente a chamada de sistema.  
Ex: `kernel/sys.c`
4. Crie uma aplicação para testar a chamada de sistema

## Exercício

- Implementar uma chamada de sistema que retorne
  - O identificador do processo: pid
- O processo em execução é dado pela variável global *current*