

# Sistemas Operacionais II

## *Aula 4 - Sinais*

Autor: Renê de Souza Pinto

Orientação: Prof. Dr. Francisco José Monaco

`rene@grad.icmc.usp.br`, `monaco@icmc.usp.br`

Universidade de São Paulo

Instituto de Ciências Matemáticas e Computação - ICMC

Escola de Engenharia de São Carlos - EESC

# Sumario

- Introdução aos sinais
- Interceptar Sinais
- Enviando Sinais
- Trabalho 1

# Sinais

- Sinais são notificações geradas para os processos, que podem interceptá-las ou não.
- Exemplo: Quando pressionamos Ctrl+Z no shell, o sinal ***SIGTSTP*** é enviado ao processo corrente, parando a execução do mesmo.

# Interceptar Sinais

- Um processo pode associar uma função Y (manipuladora) para um determinado sinal, ou seja, pode informar ao S.O. que ao chegar um sinal X, a função Y deve ser executada.
- *syscalls* utilizadas: *signal()*<sup>a</sup> e *sigaction()*

- ```
int sigaction(int signum, const struct sigaction *act,
              struct sigaction *oldact);
```

```
typedef void (*sighandler_t)(int);
sighandler_t signal(int signum, sighandler_t handler);
```

---

<sup>a</sup>Obsoleta, *sigaction* deve ser utilizada.

# Codigo exemplo

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

void catch(int signum);

int main(int argc, char **argv)
{
    struct sigaction act;

    act.sa_handler = catch;
    act.sa_restorer = NULL;
    sigaction(SIGTSTP, &act, NULL);

    printf("\nAguardando sinal...\n"); fflush(stdout);
    while(1);

    return(0);
}
```

# Codigo exemplo

```
/* Funcao Y */  
void catch(int signum)  
{  
    if(signum == SIGTSTP) {  
        printf(" Sinal interceptado =:0\n"); fflush(stdout);  
    }  
}
```

# Enviando Sinais

- Sinais são enviados aos processos através da syscall *kill()*

- ```
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```

# Codigo exemplo - kill

```
/* ... */

pid_t child;

if( !(child = fork()) ) {
    execve("/usr/bin/find", NULL, NULL);
} else {
    kill(child, SIGTSTP);
}

/* ... */
```



# Trabalho 1

# Trabalho 1

- Toda a teoria necessária para o desenvolvimento de um Shell funcional foi coberta. Agora é com você!

# Trabalho 1

- Toda a teoria necessária para o desenvolvimento de um Shell funcional foi coberta. Agora é com você!
- Objetivo:
  - Desenvolver um shell que tenha suporte a controle de processos (Ctrl+Z, Ctrl+C, etc), redirecionamentos de entrada e saída e múltiplos pipes.

Confira a especificação completa do trabalho no site da disciplina (GoogleCode).



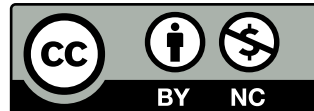
**Bom Trabalho!**

# Bibliografia

## Referências

[1] Linux manual pages

# Licença



Este documento é licenciado sob a  
Creative Commons Atribuição-Usó Não-Comercial 2.5  
Brasil License.