# MC514
# Sistemas Operacionais:
# Teoria e Prática

1s2006

## Tratamento de sinais

# Como tratar erros de execução?

```
FILE *file = fopen ("arq.txt","r");
```

- Valor de retorno indica se a execução foi bem sucedida:

  > Upon successful completion fopen returns a FILE pointer. Otherwise, NULL is returned and the global variable errno is set to indicate the error.

- Veja o manual: fopen, errno e perror
- Veja o código: fopen.c

# Como tratar erros deste tipo?

```
int *px = (int*) 0x01010101;
*px = 0;
```

- Programa recebe um sinal SIGSEGV
- O comportamento padrão é terminar o programa
- Veja o código: segfault1.c (use o gdb!)

# E erros deste tipo?

```
int i = 3/0;
```

- Programa recebe um sinal SIGFPE
- O comportamento padrão é terminar o programa
- Veja o código: div0.c (use o gdb!)

# Sinais

- Indicam a ocorrência de condições excepcionais
- Tipos de sinais
  - Divisão por zero
  - Acesso inválido à memória
  - Interrupção do programa
  - Término de um processo filho
  - Alarme
- Existem sinais síncronos e assíncronos

# Alarme

## Exemplo de sinal assíncrono

```
unsigned int alarm(unsigned int seconds);
```

- Envia um sinal do `SIGALRM` para o processo após alguns segundos.
- Veja o código: alarm1.c

# Como ignorar um sinal?

- É possível ignorar SIGALRM?

  ```
  signal(SIGALRM, SIG_IGN);
  ```

  Veja o código: alarm2.c

- É possível ignorar SIGSEGV?

  ```
  signal(SIGALRM, SIG_IGN);
  ```

  Veja o código: segfault2.c

# Como tratar um sinal?

- Rotina `signal` permite alterar o comportamento do programa em relação ao recebimento de um sinal específico.

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

# Como tratar SIGSEGV?

- Devemos escrever um tratador

```
void trata_SIGSEGV(int signum) {
  /* ... */
}
```

- e instalá-lo

```
signal(SIGSEGV, trata_SIGSEGV);
```

- Veja o código: segfault3.c (use o gdb!)

# Como recuperar o tratador padrão?

```
signal(SIGALRM, SIG_DFL);
```

- É possível fazer isso a partir do programa principal

  Veja o código: alarm3.c

- ou a partir do próprio tratador.

  Veja os códigos: alarm4.c e segfault4.c

# Um comentário sobre portabilidade

The original Unix signal() would reset the handler to SIG_DFL, and System V (and the Linux kernel and libc4,5) does the same. On the other hand, BSD does not reset the handler, but blocks new instances of this signal from occurring during a call of the handler. The glibc2 library follows the BSD behaviour.

# Problemas de consistência

- Um tratador de sinais pode encontrar dados "inconsistentes".
- Veja o código: consistencia.c
- Quais funções podem ser invocadas a partir de um tratador de sinais?

# Controle de execução

- SIGKILL: encerra a execução.
- SIGTERM: encerra a execução,mas um tratador pode ser invocado.
- SIGSTOP: interrompe a execução.
- SIGTSTP: interrompe a execução, mas um tratador pode ser invocado.
- SIGCONT: continua a execução

- Veja o código: sigcont.c