# Processos e Threads

## Tratamento de sinais

# Como tratar a morte de um filho?

```
if (fork() != 0) /* Processo pai */
   if (wait(NULL))
    printf("Meu filho morreu\n");
```

- Processo pai fica bloqueado até que um filho morra.
- Veja o código: ../proc/wait1.c

# Como tratar a morte de um filho?

```
if (fork() != 0) /* Processo pai */
   while (waitpid(-1,NULL, WNOHANG) == 0) {
      printf("Meu filho ainda não morreu\n");
      faz_alguma_coisa();
   }
}
```

- Processo pai faz verificações periódicas enquanto o filho não morre.
- Veja o código: ../proc/waitpid1.c

# Como tratar a morte de um filho?

```
void trata_SIGCHLD(int signum) {
  int pid;
  pid = wait(NULL);
  printf("Meu filho %d morreu.\n", pid);
}
```

- Sinal SIGCHLD é enviado quando um filho morre.
- Veja o código: sigchld1.c

# Como bloquear sinais

Trabalha-se com um conjunto de sinais

```
sigset_t set;
```

sobre o qual as seguintes operações são possíveis:

- `int sigemptyset (sigset_t *SET);`
- `int sigfillset (sigset_t *SET);`
- `int sigaddset (sigset_t *SET, int SIGNUM);`
- `int sigdelset (sigset_t *SET, int SIGNUM);`

# Como bloquear sinais

```
int sigprocmask (int HOW,
                 const sigset_t *restrict SET,
                 sigset_t *restrict OLDSET)
```

- SIG_BLOCK: bloqueia os sinais no conjunto set, adicionando-os à máscara atual.
- SIG_UNBLOCK: desbloqueia os sinais no conjunto set, removendo-os da máscara atual
- SIG_SETMASK: substitui a máscara atual.
- Máscara anterior é retornada em OLDSET.

# Implementando sleep()

## Funciona sempre?

```
int sleep(int nseg) {
   /* Bloqueia todos os sinais
      exceto SIGALRM */
    alarm(nseg);
    pause();  /* Bloqueia execução
                 até a chegada de um sinal */
   /* Restaura máscara anterior */
}
```

- Veja o código: sleep.c

# Implementando sleep()

## Funciona sempre?

```
int sleep(int nseg) {
   /* Bloqueia todos os sinais */
    alarm(nseg);
   /* Desbloqueia SIGALRM em mask */
    sigsuspend(&mask);  /* Bloqueia execução,
                           instala mask e
                           aguarda um sinal */
   /* Restaura máscara anterior */
}
```

- Veja o código: sigsuspend.c

# Como tratar a morte de um filho?

- Suponha que o processo pai quer gerar todos os filhos antes de saber das mortes.

```
/* Bloqueia SIGCHLD */
 gera_filhos();
/* Desbloqueia SIGCHLD */
/* Aguarda mortes */
```

- Será que usando o mesmo tratador do código sigchld1.c todas as mortes serão percebidas?
- Veja o código: sigchld2.c

# Como tratar a morte de um filho?

```
void trata_SIGCHLD(int signum) {
  int pid;
  while ((pid = waitpid(-1,NULL,WNOHANG)) > 0) {
    printf("Meu filho %d morreu.\n", pid);
    n_filhos++;
  }
}
```

- Mais de um filho pode ter morrido enquanto o sinal não foi tratado.
- Veja o código: sigchld3.c

# Duelo entre pai e filho

- Pai envia SIGTERM para o filho
- Filho envia SIGTERM para o pai
- Ambos devem morrer
- Veja os códigos: duelo1.c e duelo2.c

# Tratadores encadeados

- Um sinal pode ser tratado durante o tratamento de outro sinal
- Veja o código: encadeados.c
- Como tentar bloquear isto?
- Veja o código: encad-bloq1.c

# sigaction()

```
int sigaction(int signum,
              const struct sigaction *act,
              struct  sigaction *oldact);
struct sigaction {
  void (*sa_handler)(int);
  sigset_t sa_mask;
  /*   */
};
```

- Estabele uma função e uma máscara para ser usada no momento do tratamento do sinal.
- Veja o código: encad-bloq2.c

# Como interromper e retomar a execução de um processo?

- SIGSTOP
- SIGTSTP
- SIGCONT

- Veja o código: sigcont.c

# Como depurar um processo filho?

## Primeira tentativa

- Após o fork() o processo filho pode interromper seu processamento via raise(SIGSTOP);
- O gdb pode depurar um processo que já está rodando via comando attach
- Colocamos um breakpoint adequado no processo filho
- Enviamos um sinal SIGCONT para o processo filho
- Veja o código: attach.c

# Como depurar um processo filho?

## Segunda tentativa

- Após o fork() o processo filho pode interromper seu processamento via sigsuspend().
- Processo filho aguarda SIGUSR1
- Usuário envia SIGUSR1
- Veja o código: attach_SIGUSR1.c