Sinais

- O mecanismo *signal* (“sinal”) é uma forma de enviar **assincronamente** notificações aos processos.
  - Funciona como um “aviso”, para o processo, de que algo aconteceu.
  - Aviso pode ser recebido a qualquer altura, provocando uma interrupção (temporária ou não) na execução do programa.
  - Programação orientada ao evento.

Sinais

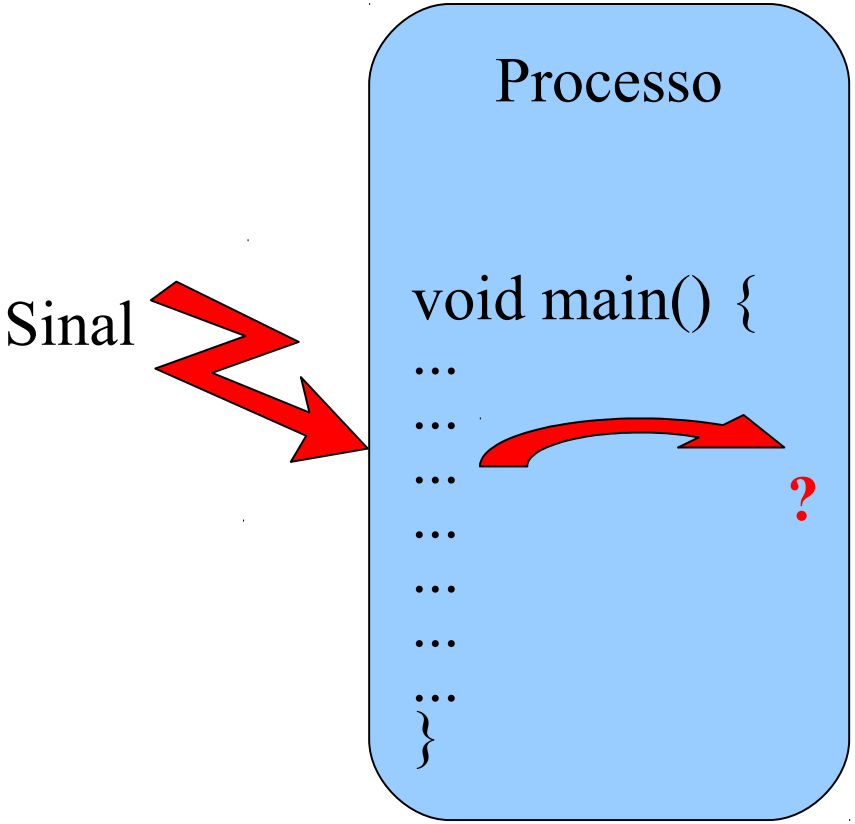
- Os sinais podem ser gerados por:
  - Programas em funcionamento.
  - Pelo próprio núcleo do sistema operativo.

- Exemplos:
  - A combinação de teclas ctrl+c e o comando `kill` utilizam o mecanismo de sinais.
  - Sinais (pré-definidos) que o sistema operativo envia aos programas em execução em caso de erro:
    - SIGFPE (floating point exception),
    - SIGSEGV (Invalid memory reference → segmentation fault)

Envio de sinais

- *Shell*
  - `kill [ -signal | -s signal ] pid`
  - Exemplo: `kill -9 12345`
- Programação C
  - `int kill(pid_t pid, int sig);`
  - `int raise(int sig); <=> kill(getpid(), sig);`
- Exceptuando o caso do *super-user (root),* o sistema operativo só permite o envio de sinais entre processos do mesmo utilizador.

Receção de sinais num processo



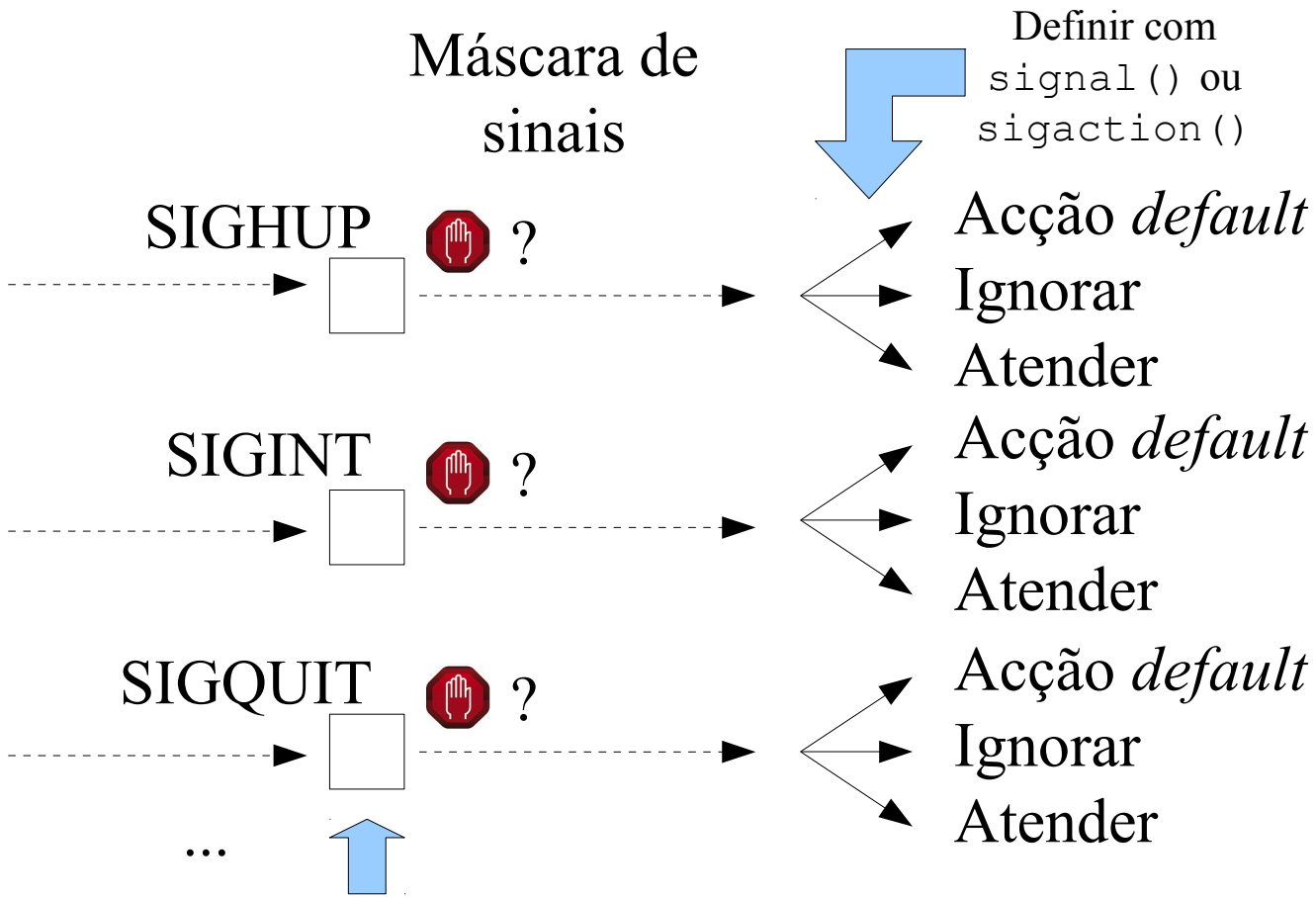
Receção de sinais

- O que acontece ao programa quando recebe um sinal?
  - Dependendo do tipo de sinal:
    - O processo termina (o caso mais frequente)
    - Não acontece nada, o sinal é ignorado (e.g., SIGCHLD).
    - Casos especiais: SIGSTOP, SIGCONT
  - À exceção do SIGKILL e SIGSTOP, é possível mudar individualmente a disposição do processo para cada sinal:
    - O sinal pode ser ignorado.
    - O sinal pode provocar a execução imediata de uma função (interrompendo a função em execução).

Bloqueio de sinais

- É possível a adiar a receção dos sinais enviados a um processo.
- Cada sinal pode ser bloqueado e desbloqueado conforme seja conveniente para o programa.
  - Caso o sinal seja recebido enquanto está bloqueado, o sinal fica pendente (*pending*), i.e., em espera, até que seja desbloqueado (man 7 signal):
    - *" A signal may be blocked, which means that it will not be delivered until it is later unblocked. Between the time when it is generated and when it is delivered a signal is said to be pending."*
- Cada processo tem uma "máscara de sinais" (*signal mask*) que, em cada altura, indica o conjunto de sinais que ficarão em espera (*pending*) caso sejam enviados ao processo. A máscara de sinais pode ser alterada durante a execução do processo conforme desejado.

Tratamento de sinais a nível do processo



Buffer para 1 ocorrência do sinal (sinal pendente).

Sinais mais comuns

SIGINT - Interrupt from keyboard (CTRL+C)
SIGQUIT - Quit from keyboard (CTRL+Q)
SIGTERM - Termination signal
SIGKILL - Kill signal
SIGABRT - Abort signal from abort(3)
SIGCONT - Continue if stopped
SIGSTOP - Stop process
SIGTSTP - Stop typed at tty (CTRL+Z)
SIGHUP - Hangup on controlling terminal or death of controlling process
SIGCHLD - Child stopped or terminated

SIGPIPE - Broken pipe: write to pipe with no readers
SIGALRM - Timer signal from alarm(2)

SIGUSR1 - User-defined signal 1
SIGUSR2 - User-defined signal 2

SIGSEGV - Invalid memory reference
SIGFPE - Floating point exception

Conjuntos de sinais (*sigset_t*)

- `int sigemptyset(sigset_t *set);`
  - Faz com que **set* seja um conjunto vazio.
- `int sigfillset(sigset_t *set);`
  - Adiciona todos os sinais do sistema ao conjunto **set*.
- `int sigaddset(sigset_t *set, int sig);`
  - Adiciona o sinal sig ao conjunto **set*.
- `int sigdelset(sigset_t *set, int sig);`
  - Remove o sinal *sig* do conjunto **set*.
- `int sigismember(sigset_t *set, int sig);`
  - Retorna 1 se o sinal *sig* faz parte do conjunto **set*.

Bloqueio de sinais

- `int` **`sigprocmask`**`(int` *`how`*`, sigset_t *`*`new_set`*`, sigset_t *`*`old_set`*`);`
  - Usada para alterar ou consultar a máscara de sinais do processo.
  - parâmetro *how*:
    - SIG_BLOCK - Bloqueia todos os sinais indicados em new_set (i.e., adiciona-os à mascara de sinais do processo).
    - SIG_UNBLOCK – Desbloqueia todos os sinais indicados em *new_set* (i.e., remove-os da máscara de sinais do processo).
    - SIG_SETMASK - todos os sinais indicados em *new_set* são bloqueados. Os restantes são desbloqueados.
  - Se *new_set* for NULL, nenhuma alteração é feita.
  - Se *old_set* for diferente de NULL, é retornada (em *old_set*) a atual máscara de sinais do processo (i.e., previamente à execução da função).
-

Sinais pendentes

- `int sigpending(sigset_t *set);`
  - Preenche `*set` com o conjunto de sinais recebidos pelo processo mas que ficaram bloqueados.

Configuração da resposta ao sinal

- POSIX:
  - `int sigaction(int signal_number, struct sigaction *new_handler, struct sigaction *old_handler);`
    -
```
struct sigaction {
    void (*sa_handler)(int);
    sigset_t sa_mask; /* new signal mask*/
    int sa_flags; /* options */
};
```
- Formato tradicional do System V (ainda suportado):
  - `sighandler_t signal(int signum, sighandler_t handler);`
    - `typedef void (*sighandler_t)(int);`

Configuração da resposta ao sinal

- Função **`sigaction`**
  - Configura o comportamento do processo em relação ao sinal indicado.
  - Basta chamar uma vez para cada tipo de sinal
    - A configuração pode manter-se independentemente das vezes que o sinal é recebido.

- Campo `sa_handler`
  - Função definida pelo utilizador.
  - SIG_IGN (o sinal é ignorado)
  - SIG_DFL (volta a estabelecer a acção *default*)

sigaction

- Campo `sa_mask` – indica os sinais que deverão passar a ser bloqueados durante o atendimento do sinal em questão.

- Campo `sa_flags`
  - SA_RESETHAND – após a receção de um sinal, volta a assumir o comportamento *default* (SIG_DFL)
  - SA_NODEFER – o sinal pode interromper a sua própria rotina de atendimento.
  - SA_RESTART – permite que funções do sistema (e.g., leitura de um ficheiro) interrompidas possam recomeçar, em vez de gerar um erro.
  - SA_NOCLDSTOP (SIGCHLD) – processo pai não é avisado das ocorrências de SIGSTOP/SIGCONT nos processos filho.
  - SA_NOCLDWAIT (SIGCHLD) – processos filho não passam pelo estado *zombie.*

Exemplo 1

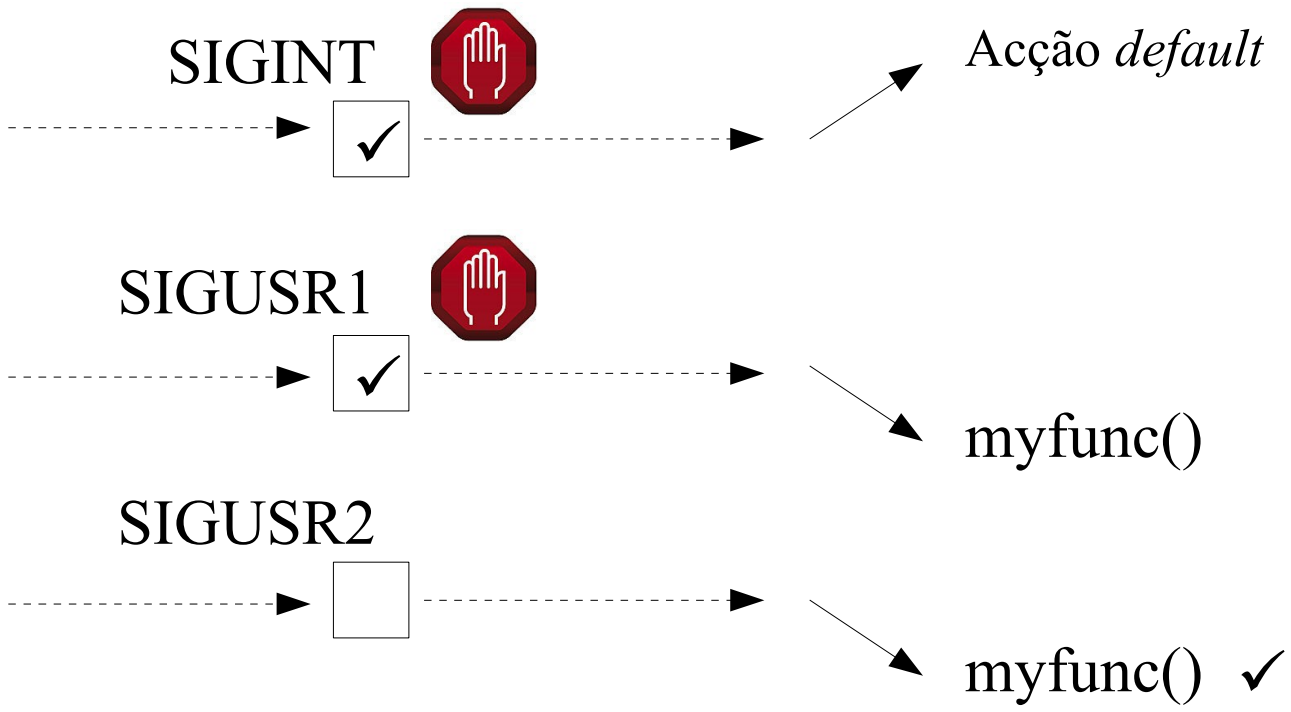
- Ignorar um sinal: `signal(nome_do_sinal, SIG_IGN)`
- Processo ignora SIGINT e atende SIGTERM:

```
void myhandler(int signum) {
  //faz alguma coisa...
}

int main() {
  signal(SIGINT, SIG_IGN);

  struct sigaction act;
  act.sa_handler = myhandler;
  act.sa_flags = 0;
  sigemptyset(&(act.sa_mask));
  sigaction(SIGTERM, &act, NULL);
  (...)
```

Exemplo 2: como obter a seguinte configuração?



Exemplo (solução)

```
int main() {
  struct sigaction act;
  act.sa_handler = myfunc;
  act.sa_flags = 0;
  sigemptyset(&(act.sa_mask));
  sigaction(SIGUSR1, &act, NULL);
  sigaction(SIGUSR2, &act, NULL);

  sig_set s;
  sigemptyset(&s);
  sigaddset(&s, SIGINT);
  sigaddset(&s, SIGUSR1);
  sigprocmask(SIG_BLOCK, &s, NULL);

  sigemptyset(&s);
  sigaddset(&s, SIGUSR2);
  sigprocmask(SIG_UNBLOCK, &s, NULL);
```

Funções bloqueantes

- `int sigsuspend(const sigset_t *mask);`
  - Temporarily replaces the signal mask of the calling process with the mask given by mask and then suspends the process until delivery of a signal whose action is to invoke a signal handler or to terminate a process.

- `int pause(void);`
  - Causes the calling process (or thread) to sleep until a signal is delivered that either terminates the process or causes the invocation of a signal-catching function.

Notas adicionais

- Para certas ocorrências (SIGFPE, SIGILL, SIGSEGV), o processo fica numa situação instável e, mesmo sendo possível atender o sinal, a única solução é terminar o processo.

- Os sinais SIGKILL e SIGSTOP não podem ser apanhados (i.e., provocar a execução de uma função), ignorados nem bloqueados.

- As páginas “man” contêm informação detalhada sobre o funcionamento dos sinais: **man 7 signal**.