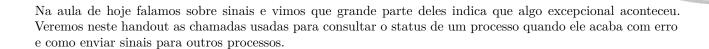
# Insper



Igor Montagner



# Parte 1 - recuperando informações de erros usando wait

Anteriormente vimos que ao chamar wait(&status); guardamos informações sobre o fim do processo filho na variável status. Nos outros exercícios olhamos para os casos em que WIFEXITED(status) == 1.

Todo término inesperado de um programa é feito usando um sinal. Ao acessar informações em um local de memória não mapeado para o nosso processo ele recebe o sinal SIGSEGV. Ao executar uma divisão por zero ele receberá o sinal SIGFPE. Logo, nestes casos WIFSIGNALED(status) == 1 e podemos pegar o número do sinal usando a macro WTERMSIG(status).

**Exercício** Crie um programa que cria um processo filho e o espera. O processo filho deverá realizar uma divisão por 0 logo após o fork. No proceso pai, após o wait mostre no terminal as seguintes expressões booleanas:

- WIFEXITED(status)
- WIFSIGNALED(status)
- WTERMSIG(status)

Exercício: mostrar o número do sinal não é muito útil. Pesquise sobre a chamada strsignal e use-a para mostrar uma mensagem descritiva de qual sinal foi recebido no exercício acima.

#### Parte 2 - envio de sinais via terminal

Além de erros e exceções, sinais também são usados para avisar de mudanças no sistema, sejam elas iniciadas pelo usuário ou por outros processos. A sequência de exercícios abaixo é um experimento de envio de sinais.

Exercício: Crie um programa que faz um fork(). O processo pai deverá esperar o filho acabar e mostrar suas informações de finalização. O processo filho mostra seu pid no terminal e entra em loop infinito.

Claramente nem o pai nem o filho terminam no exemplo acima. Porém, se o filho terminar o pai termina também! O sinal **SIGKILL** é usado para terminar forçadamente um processo e ele pode ser enviado por qualquer outro processo do mesmo usuário (ou o *root*).

O envio de sinais é feito usando a chamada kill. Assim como outras chamadas de sistema, kill também é um programa de linha de comando.

Exercício: Use a ferramenta de linha de comando kill para enviar o sinal SIGKILL para o processo filho. Se precisar, consulte a documentação usando man 1 kill.

Isto deve ter feito o pai finalizar e mostrar a informação de que o processo filho foi finalizado. Note que o pai tem direito a saber o sinal usado para finalizar um filho:

**Exercício**: Envie o sinal SIGINT para seu processo filho e verifique que o processo pai mostra o número correto.

### Parte 3 - envio de sinais em um programa

O programa kill é apenas um casquinha em volta de sua chamada de sistema.

Exercício: Veja a documentação da chamada de sistema (em C) no manual man 2 kill

**Exercício**: Modifique seu exercício da parte anterior para que o processo pai espere 10 segundos e envie um **SIGKILL** para o filho. Agora os dois processos acabam? As informações de finalização no pai condizem com o sinal enviado.

Exercício: execute man 7 signal para uma lista de todos os sinais existentes. Teste outras combinações com kill e seu programa do exercício anterior.

Até um momento usamos a chamada wait para esperar um processo filho acabar. Porém, não sabemos ainda checar se um processo filho acabou! Podemos fazer isso com a função waitpid, que recebe o pid\_t do processo filho, e permite esperar ou não pela finalização dele.

Exercício: pesquise no manual pela flag WNOHANG da chamada waitpid

Exercício: modifique seu exercício acima para que o processo pai só envie o sinal se o processo filho ainda estiver executando.

## Exercícios avançados

Exercício: Ao apertarmos Ctrl+C no exercício acima, são finalizados

- a) somente o processo pai
- b) somente o processo filho
- c) ambos.

Use http ou programa similar para verificar isso.

Podemos **ignorar** a chegada de um sinal usando a função signal!

Exercício: pesquise como usar a função acima para ignorar um sinal.

**Exercício**: modifique seu programa da parte 3 para que somente o processo filho seja terminado ao pressionar Ctrl+C no terminal.

**Exercício**: pesquise o que faz a chamada alarm no manual. Como você poderia usá-la para simplificar o programa da parte 3?