Instituto Superior de Engenharia de Coimbra
Departamento de Engenharia Informática e de Sistemas

Sistemas Operativos 2021/2022

Ficha Nº 4 Sinais

Tópicos ->
- Sinais – Comandos UNIX para sinais.
- API UNIX de sinais. *signal*, *kill*, *sigaction*, *sigqueue*
- Cenários exemplo de uso de sinais. Alertas, Coordenação e temporização

Exercícios

Exercício 1: Mostra o funcionamento básico de sinais em C e o envio de sinais em linha de comandos.

Exercício 2: Foca o uso de sinais para construir temporizadores.

Exercícios 3: Foca o uso de sinais com passagem de informação muito básica entre processos.

Exercícios 4,5: Mostram o API mais antigo para sinais: *signal* /*kill*.

1. Escreva um programa com as seguintes características:

- Pede o nome ao utilizador e responde com a mensagem olá *nome-introduzido*. Se escrever "sair", o programa termina.
- Sempre que o utilizador efectua ^C, apresenta a mensagem "ai" no ecrã. Ao 5º ^C, o processo termina com a mensagem "ok, pronto".

Nota: sempre que a consola detecta um ^C, envia o sinal SIGINT aos processos do grupo que está a correr em *foreground*.

Deve usar as funções do API *sigaction*/*sigqueue* em vez de *signal*/*kill*.

a) Faça o programa e teste-o. Verifique o que acontece se efectuar ^C a meio da introdução de caracteres: os caracteres já introduzidos perdem-se? Explique o que acontece.

b) Teste o programa abrindo um segundo terminal e enviando o sinal SIGINT ao processo com o comando ***kill*** (***man kill*** para os pormenores). Verifique o que acontece se enviar o sinal a meio da introdução de caracteres: os caracteres já introduzidos perdem-se?

c) Modifique o programa de forma a usar o sinal SIGUSR1.

O código deste exercício pode ser útil mais adiante. Não perca os ficheiros de código fonte.

Objetivos do exercício - Sinais
- Conceito e funcionamento de sinais em Unix.
- Mostrar como se podem usar sinais em simultâneo com operações de leitura.
- Funções *sigaction*, *sigqueue*. Comando *kill*.

2. Escreva um programa que permita ao utilizador treinar o cálculo mental. O funcionamento deste programa é o seguinte:

- De N em N segundos sorteia dois números inteiros entre 0 e 100 e apresenta-os no ecrã. O utilizador deve introduzir o valor que corresponde à soma destes valores.
- O primeiro valor de N é 20. O valor N diminui em 1 a cada novo número sorteado.
- Ao segundo valor errado ou valor não introduzido a tempo, o jogo termina. A pontuação é o número de valores certos introduzidos a tempo.

Nota: o programa não está adormecido durante os N segundos. Bem pelo contrário, o programa está em funcionamento a ler os valores que o utilizador está a introduzir no teclado. A resolução deste exercício não envolve nem ***sleep*** nem ***pause***.

Objetivos do exercício - Temporização com sinais – alarm
- Sinal SIGALRM. Construção de temporizadores com alarm e SIGALRM
- Funções *sigaction*, *sigqueue*

3. Escreva dois programas que permitam a vários utilizadores adivinharem um número que foi sorteado. Os programas são o "sorteia" e o "adivinha", os quais fazem:

- ***sorteia***. Este programa imprime no ecrã o seu PID. Depois entra em ciclo. Em cada iteração sorteia um número entre 1 e 100 e aguarda que lhe sejam *entregues* números via sinais. A cada número recebido o programa deve imprimir no ecrã o número recebido e o PID de quem o enviou, e também a mensagem "demasiado grande", "demasiado pequeno", "acertou", conforme o caso. Sempre que o número recebido é igual ao que foi sorteado, o programa sorteia um novo número. O programa termina com ^C, imprimindo o número de vezes que o número foi adivinhado.
- ***adivinha***. Este programa é lançado com um número **p** como argumento na linha de comandos. O programa imprime o seu PID no ecrã e depois entra em ciclo. Em cada iteração pede um número ao utilizador e envia esse número ao processo **p** via sinais. O programa termina quando o número introduzido é 0.

Notas

- Não há "vez do jogador": cada jogador tenta um número quando quiser.
- Inicialmente não haverá feedback do programa sorteia para o programa adivinha. Não há pontuação nem tempo máximo de resposta. Cada programa é lançado a partir de um terminal diferente.

a) Faça os dois programas de acordo com as características descritas acima. Use o comando ***man*** para descobrir como pode gerar valores aleatórios com as funções rand e srand. Testes os programas com um "*sorteia*" e dois "*adivinha*".

b) Modifique os programas sorteia e adivinha de forma a haver um feedback do sorteia para o cliente: a cada número recebido, o sorteia envia de volta (ao adivinha respectivo) o valor: 1 ("demasiado grande"), -1 ("demasiado pequeno"), 0 ("acertou"). O programa adivinha imprime estas mensagens no ecrã. O mecanismo a usar é apenas sinais.

c) Modifique o programa *sorteia* de forma a libertar a consola quando é executado (tal como se o tivesse lançado com "&").

d) Modifique o programa *sorteia* de forma a detectar se já estava a correr. Se estiver, sai simplesmente, ficando a apenas a executar o primeiro. Existem várias formas de conseguir esta funcionalidade. Neste momento, a mais fácil é o uso um ficheiro temporário.

e) (TPC) Modifique o programa "sorteia" de forma a memorizar o PID de todos os processos "cliente" que vão interagindo com ele. Quando um processo "cliente" termina envia um sinal com o valor 0 que indicará ao "sorteia" para o "esquecer". O "sorteia" deverá então ter esta nova funcionalidade:

- Quando alguém acerta no número, avisa todos os "adivinha". Os "adivinha" colocarão um aviso no ecrã, para que o utilizador possa recomeçar a sua estratégia.
- Quando o "sorteia" sai, avisa todos os clientes.

Todo o exercício, incluindo a alínea e, deve ser resolvido recorrendo apenas a sinais.

Objetivos do exercício - Sinais
- Consolidação de matéria de sinais.

4. Repita o exercício 1 usando *signal*/*kill* em vez de *sigaction*/*sigqueue*.

Objetivos do exercício - Sinais - API *signal* / *kill*.
- Tomar contacto com o API mais antigo de sinais: signal e kill.

5. Repita o exercício 2 usando *signal*/*kill* em vez de *sigaction*/*sigqueue*.

Objetivos do exercício - Sinais - API *signal* / *kill*.
- Tomar contacto com o API mais antigo de sinais: signal e kill.