Duração: 2h

I

Responda de forma clara e sucinta (não mais de 10 linhas legíveis) às questões seguintes. Pense antes de escrever! Obrigado.

1 Como sabe, um *fork* cria um processo que é praticamente idêntico seu criador: executa o mesmo programa e tem uma cópia das variáveis do pai no momento do *fork*. Mostre como o sistema operativo consegue manter os dois processos simultaneamente em memória, garantindo que cada processo só acede às suas variáveis.

2 Um arquivo de impressões digitais como o descrito no Grupo II pode ser a base para um sistema de controlo biométrico de acessos. Supondo que cada ficheiro de dados é pequeno (16 KB) e que se prevê um universo de 1000 utilizadores, identifique a carga de CPU, I/O e memória que o programa concorrente do Grupo II coloca. Acha que um portátil com linux seria má ou boa escolha para esta aplicação? Justifique. Note bem: CPU, I/O e memória.

3 Suponha que lhe pediram para simular por software um sistema RAID 1 com dois discos. Comece então por explicar o que é e porque se usa RAID 1. De seguida, apresente o pseudo código de sincronização de escritas concorrentes nos discos. Só tem de escrever 2 funções, uma que inicia cada escrita e espera passivamente pelo seu fim e outra que é invocada quando qualquer dos discos termina a operação de escrita. Pretende-se que use as operações `cria_semaforo(valor_inicial)`, `P` e `V`.

II

Dado um arquivo de impressões digitais previamente recolhidas de várias pessoas, pretende-se identificar de que pessoa é uma nova impressão digital. Suponha a existência de um programa `matcher`, que recebe como argumentos os nomes de dois ficheiros das duas impressões digitais a comparar, e escreve no *standard output* os dígitos de um inteiro, de 0 (não condizem) a 255 (correspondência perfeita). Implemente um programa que recebe como argumentos os nomes de ficheiros das impressões em arquivo e da nova impressão a comparar, faça comparação concorrente, e escreva o nome do ficheiro da impressão em arquivo que melhor corresponde (assuma que não há empates). O programa deve imprimir imediatamente o resultado e parar todo o processamento em curso caso seja encontrada uma correspondência perfeita.

III

Considere um programa `CTRL` que monitoriza a actividade de um conjunto de processos a executar no mesmo computador. Cada processo monitorizado escreve, de 3 em 3 segundos, "OK" para o seu standard output enquanto se encontra activo. Apresente o programa `CTRL` que recebe como argumentos um número arbitrário de programas (`CTRL p1 p2 ... pn`) que deverá colocar em execução e monitorizar. Se um destes processos demorar mais de 3 segundos sem escrever OK, deve ser terminado e o tempo aproximado em que esteve activo reportado para o standard output do `CTRL`. Logo que todos os processos tenham sido terminados, `CTRL` termina também a sua execução.

Algumas chamadas ao sistema relevantes

Processos

- int read(int fd, void *buf, size_t count);