MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Teste

26 de maio de 2017 — Duração: 2h

I

Responda às perguntas seguintes com *palavras suas* e de forma *sucinta*: Não exceda o limite de 10 a 15 linhas legíveis em cada.

1 Praticamente todos os dias surgem notícias de ataques a sistemas informáticos e da necessidade de protecção. Ora sendo o sistema operativo um gestor de recursos, também lhe cabe a tarefa de fornecer mecanismos para proteger esses recursos. Escolha três recursos estudados nas aulas teóricas e explique os respectivos mecanismos de protecção. Assuma que se trata de um sistema com Linux.

2 Ao programar e testar alguns exercícios propostos nos guiões das aulas práticas certamente se terá deparado com situações de *Deadlock*. Dê um ou mais exemplos **seus** (reais ou imaginados por si) que mostrem com precisão o problema e a forma de o evitar. Deve dar uma resposta muito concreta.

3 Leia com atenção o enunciado do Grupo II. Acha que para resolver esta questão seria boa ideia usar uma estratégia de escalonamento *ROUND-ROBIN* nos processos passados como argumentos (no exemplo, ls df find). Justifique a sua resposta. Mostre que percebeu a pergunta (e a resposta).

II

Pretende-se escrever o programa "paginas", que recebe como argumentos os nomes de vários executáveis e que faz com que o output desses programas apareça alternadamente, página por página, de 10 linhas cada uma. Por exemplo, correndo "paginas ls df find", deverão ser impressas 10 linhas do ls, depois 10 linhas do df, depois 10 linhas do find, depois 10 linhas do ls e assim sucessivamente até que os vários programas terminem. A ausência de output de um dos programas não deverá impedir a impressão das linhas dos outros.

III

Considere que se pretende fazer uma pesquisa concorrente para determinar a existência de determinado padrão em linhas de texto, recebidas no stdin. Assuma a existência dois programas, `filtro` e `existe` que, encadeados, permitem fazer a pesquisa. O primeiro descarta as casos mais óbvios que lhe chegam ao seu stdin, deixando passar os restantes para o seu stdout. O segundo termina com valor de exit de 1 caso receba uma linha com o padrão no seu stdin, ou com 0, caso isso não aconteça, até EOF. Escreva um programa que receba um número n como argumento, e faça a pesquisa concorrente utilizando n pares de processos a executar `filtro` e `existe`. O programa deverá imprimir o resultado, logo que possível, e terminar os processos que já não estejam a faer processamento relevante.

Algumas chamadas ao sistema relevantes

Processos

- pid_t fork(void);
- void exit(int status);
- pid_t wait(int *status);
- pid_t waitpid(pid_t pid, int *status, int options);
- WIFEXITED(status);
- WEXITSTATUS(status);
- int execlp(const char *file, const char *arg, ...);
- int execvp(const char *file, char *const argv[]);

Sinais

- int read(int fd, void *buf, size_t count);
- int write(int fd, const void *buf, size_t count);
- long lseek(int fd, long offset, int whence);
- int access(const char *pathname, int amode);
- int pipe(int filedes[2]);
- int dup(int oldfd);
- int dup2(int oldfd, int newfd);