

## Sistemas Operativos I

Exame de recurso<sup>1</sup>

17 de Fevereiro de 2007

Duração: 2h30m

### I

1. É sabido que a gestão de memória em sistemas de memória real multiprogramados é pouco eficiente. Explique em que consiste esse desperdício de recursos e mostre como a memória virtual poderá aliviar ou eliminar essa falta de eficiência.
2. Uma dos objectivos a atingir pela gestão de memória é a *protecção*. Explique o que significa, e procure mostrar como pode ser conseguida em sistemas de memória virtual.
3. Uma base de dados é normalmente sujeita a acessos concorrentes que poderão causar problemas se duas ou mais actividades tentarem aceder à mesma tabela ou registo ao mesmo tempo (*race conditions*). À primeira vista, uma forma de evitar esse problema seria serializar os acessos usando uma estratégia de escalonamento FIFO. Concorde com esta solução? Qual a melhor opção, caso não concorde com FIFO? Justifique. Por outras palavras, se ao instalar o SO da máquina que suporta a base de dados lhe aparecesse uma caixa a dizer escalonamento FIFO, aceitaria essa opção? Que vantagens e desvantagens teria face a outras opções dessa caixa?

### II

Codifique o programa “monitor” que executa todos os programas que lhe forem passados na linha de comando. O “monitor” afixa constantemente o número de processos filho ainda em execução e, de 5 em 5 segundos, envia a cada um destes processos em execução um sinal SIGUSR1. O “monitor” termina a sua execução assim que todos os processos o façam.

### III

Implemente o programa “logger” que deverá re-encaminhar o seu *standard input* para diversas aplicações externas através de filtros *grep* com um dado padrão. O programa em questão deverá ainda tratar graciosamente a finalização de uma ou mais aplicações externas (ou dos filtros).

```
$ logger aplicacaoA padraoA aplicacaoB padraoB ...
```

#### Protótipos das chamadas ao sistema relevantes

##### Processos

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

##### Sistema de Ficheiros

- `int open(const char *pathname, int flags, mode_t mode);`
- `int creat(const char *pathname, mode_t mode);`

- `int close(int fd);`
- `int read(int fd, void *buf, size_t count);`
- `int write(int fd, const void *buf, size_t count);`
- `int pipe(int fildes[2]);`
- `int dup(int oldfd);`
- `int dup2(int oldfd, int newfd);`

##### Sinais

- `void (*signal(int signum, void (*handler)(int)))(int);`
- `int kill(pid_t pid, int signum);`
- `int alarm(int seconds);`
- `int pause(void);`

<sup>1</sup>Cotação — 8+12