LICENCIATURA EM ENGENHARIA INFORMÁTICA
LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Exame de Recurso

13 de Junho de 2022 Duração: 2h

Por favor responda ao grupo I e a cada exercício do grupo II em folhas de teste separadas.
No grupo I espera-se que não use mais de 15-20 linhas legíveis para cada questão. Obrigado.

I

1 Explique os motivos para o algoritmo de escalonamento de processos RR – *Round Robin* continuar a ser muito popular, apesar do SRT – *Shortest Remaining Time* tipicamente conseguir menores tempos de espera na *ready queue*.

2 Explique em que consiste o *swapping* de processos, e explique porque pode ser útil num sistema que tenha memória virtual suportada por paginação.

II

1 Implemente a função `comando`, com o protótipo abaixo, que execute um programa `filtro`, sem argumentos. Esse comando deverá processar o ficheiro `entrada` no seu *standard input* e escrever no ficheiro `saida` o resultado do seu *standard output*. A função deverá retornar verdadeiro ou falso consoante `filtro` escreva ou não no *standard error*. Não utilize ficheiros auxiliares.

```
int comando(const char* filtro, const char* entrada, const char* saida);
```

2 Implemente agora um servidor acessível via *pipe com nome* que execute a referida função a pedido dos clientes. Escolha um formato de mensagem que resulte em código simples, tendo em conta que cada argumento cabe num array de 20 chars, incluíndo o terminador de *string*. Este servidor deve ir recebendo pedidos até ter 5 pedidos com os respectivos argumentos — filtro, entrada e saida — e só nessa altura invoca concorrentemente a função `comando` de cada pedido. De seguida, acrescenta ao ficheiro `comando.log` os argumentos das invocações que retornem verdadeiro (basta uma linha de texto com os argumentos enviados pelo cliente). O servidor só deverá aceitar nova ronda de pedidos quando tiver acabado de processar as 5 respostas anteriores.

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[]);
- int execve(const char *file, char *const argv[], char *const envp[]);

Sistema de Ficheiros

- int open(const char *pathname, int flags, 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);
- 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);

Sinais

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