MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Exame de Recurso

26 de Junho de 2017 Duração: 2h

I

1 Ao criar um processo em Linux, é habitual encontrar o padrão seguinte:

```
if ((p = fork()) == 0) /* filho... */ else /* pai... */
```

Explique com rigor como é possível executar simultaneamente 2 processos "iguais" sem que o filho consiga aceder às variáveis do pai (e vice-versa).

2 Explique a necessidade do fork() acima ser uma *system call* e não uma função normal.

3 Discuta em que medida um sistema de Raid 5 permite recuperar informação adulterada num ataque de "ransomware" (ficheiros deliberadamente cifrados para forçar pagamento de resgate).

II

Escreva um programa, invocado da forma `controlador <p> <c>`, para processamento de linhas de texto produzidas concorrentemente por p processos, instâncias de um programa `produtor` (que se assume existir, invocado sem argumentos), por um de c processos, instâncias de um programa `consumidor` (que se assume existir, invocado sem argumentos). Em cada momento todas as linhas produzidas devem ser processadas por um mesmo consumidor. Com o tempo deve ser feita uma escolha rotativa do consumidor corrente, devendo ser escolhido o consumidor seguinte quando o processo `controlador` receber o sinal `SIGUSR1`.

III

Escreva um programa, invocado da forma `streamer <p1> <p2> ... <pn>`, que encadeia as saídas e entradas standard dos programas `p1` a `pn`. No caso de `streamer` receber um sinal `SIGUSR1`, os programas de deverão ser terminados assegurando que qualquer informação lida por `p1` é antes processada e impressa por `pn`.

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);