LICENCIATURA EM ENGENHARIA INFORMÁTICA

Sistemas Operativos

Teste[1]

11 de Junho de 2011 Duração: 2h00m

I

1 Descreva a utilidade da chamada ao sistema "wait", e explique porque pode ser problemático um processo que corre indefinidamente (como um daemon) não a utilizar.

2 Arquitecturas modernas (e.g. Intel 64) suportam páginas com tamanhos muito maiores (e.g. 2MB) do que o normal. Descreva que vantagens tal pode oferecer, e dê um exemplo, justificando, de uma estrutura de dados apropriada para ser alocada numa região de memória baseada neste tipo de páginas.

3 Explique em que consiste um sistema de ficheiros "log-structured", bem como as suas vantagens face a um convencional. Justifique a sua resposta baseando-se na comparação com o sistema de ficheiros tradicional do Unix (com i-nodes). Admita 3 cenários, um de pesquisa de um grande volume de informação read-only (e.g consulta do número de eleitor), outro de actualização de stocks e um terceiro de criação de logs aplicacionais para efeitos de auditoria (por exemplo, registo de todos os pedidos efectuados aos servidores web).

II

Escreva em C um programa `corre`, que recebe como argumentos dois números N e S, seguidos de nomes de programas; e.g., "`corre 2 30 prg1 prg2 prg3 prg4`". O programa deverá executar os programas passados como argumento, devendo explorar concorrência, mas garantir que 1) em cada momento existam no máximo N processos a correr os programas; 2) cada programa corre no máximo S segundos.

III

Considere a existência do utilitário `oc` que procura no seu standard input o termo que lhe é passado como argumento e imprime o número de ocorrências. Pretende-se que codifique em C um programa `multi-oc` que, utilizando `oc`, faça a procura de um termo em vários ficheiros concorrentemente e imprima o número de ocorrências por ficheiro. Como exemplo, o programa é invocado por "`multi-oc sistemas ficheiro1.txt ficheiro2.txt ficheiro3.txt`" e produz o resultado: "`ficheiro1.txt: 154 ficheiro2.txt: 111 ficheiro3.txt: 0`"

Protótipos de algumas funções e 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[]);
```

Sinais

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

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 filedes[2]);
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

---

[1] Cotação — 8+6+6