LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Exame de Recurso

18 de Junho de 2019 Duração: 2h

Por favor responda a cada um dos 3 grupos em folhas de teste separadas. Obrigado.

I

1 Na estratégia de escalonamento SRT (shortest remaining time), um processo poderá ser desafectado se "chegar" outro com duração esperada menor. Explique como podem ser estimadas tais durações e, supondo um número fixo de processos em execução, descreva como poderá "chegar" tal processo.

2 Arquitecturas modernas suportam *huges pages*, páginas com tamanhos muito maiores (e.g., 2MB) do que o clássico. 1) descreva que vantagens tal pode oferecer; 2) dê um exemplo, justificando, de uma estrutura de dados apropriada para ser alocada numa região de memória baseada neste tipo de páginas; 3) se fosse mapear em memória os ficheiros do trabalho prático, diga para cada um (artigos, strings, stocks, vendas) se deveria ou não considerar o uso de *huge pages*.

II

Assuma: a existência de um programa `geturls`, que recebe uma URL como argumento, descarrega a página correspondente e produz no *stdout* as URLs nela contidos, uma por linha, cada uma escrita atomicamente num único `write`; que cada URL não excede os 100 bytes; a existência de uma função `readline`, com o mesmo protótipo de `read`.

Escreva um programa `collect` que concatene num ficheiro, cujo nome é passado por argumento, todas as URLs (incluindo repetidas) produzidas por `geturls` aplicado a cada linha (que se assume ser uma URL) do *stdin* de `collect`. Explore concorrência, mas limite a 10 o número de processos a correr `geturls` em cada momento, e dê a cada execução de `geturls` um minuto para terminar, forçando a terminação se necessário.

III

Considere um programa que lê URLs do seu stdin (uma por linha), descarrega a página Web correspondente e imprime as URLs nela contidas, cada uma escrita atomicamente num único write. Para obter todas as URLs nas páginas, sem repetições, usa-se na shell em conjunto com o programa uniq da seguinte forma:

```
$ fetchurls < seedURLs.txt | uniq
```

Escreva um programa em C com a mesma funcionalidade e fazendo uso dos programas `fetchurls` e `uniq` mas que permita ter 8 processos a descarregar páginas concorrentemente. Assuma que cada URL não excede os 100 bytes e a existência de uma função `readline`, com o mesmo protótipo de `read`.

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);