MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

Sistemas Operativos

Teste

28 de Maio de 2019 Duração: 2h

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

I

1 Explique porque o algoritmo de escalonamento *round robin* é bastante popular, em comparação com outros que podem parecer melhores (como o *shortest remaining time*, que até leva tipicamente a menores tempos de espera na *ready queue*). Descreva uma melhoria possível sobre o algoritmo básico, adequada a cenários interactivos.

2 Durante as aulas foram analisadas diversas situações em que foi possível executar, sem degradação visível de desempenho, conjuntos de programas que no total pediam centenas de vezes mais memória do que a RAM existente. Houve também casos em que a quantidade de memória pedida ao sistema operativo infelizmente atrasou muito a execução de um ou mais programas. Dê exemplos de cada uma destas situações opostas e justifique/explique porque acontecem (usando, naturalmente, os seus conhecimentos de memória virtual).

II

Suponha que um professor pretende deixar de publicar no Blackboard as classificações dos alunos inscritos na sua UC, passando a enviar por mail a cada um apenas a respectiva classificação. O referido professor já tem um ficheiro de texto devidamente formatado com <aluno> <nota>, 10 caracteres por linha, e quer agora evitar o trabalho de enviar manualmente centenas de mails! Veja o exemplo seguinte:

```
$ mail -s Sistemas_Operativos  a00000@alunos.uminho.pt
a00000 18
<ctrl-D>
```

Escreva um programa em C que leia linhas do ficheiro (passado como argumento) com os pares <aluno><nota> e envie o respectivo email, executando o comando `mail` como exemplificado acima. Espera-se uma solução concorrente mas limitando a concorrência a não mais de 10 processos simultâneos de mail.

III

Assuma a existência de um programa `patgrep` que, recebendo um padrão como argumento, escreve no *stdout* um 'X' por cada bloco de 128 bytes do seu *stdin* que contenha o padrão.

Escreva um programa `counter` que escreva no *stdout* quantos blocos de 128 bytes do seu *stdin* contêm um padrão passado por argumento. Faça uma procura concorrente, através de 8 instâncias do programa `patgrep`.

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);