LICENCIATURA EM ENGENHARIA INFORMÁTICA
LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos

Teste[1]

12 de Junho de 2015 Duração: 2h00m

I

1 Num sistema de atendimento ao público com fila única os utentes têm de esperar por ordem de chegada até que um dos balcões esteja livre, dirigindo-se então para esse balcão. Não há nenhum funcionário a controlar, são os próprios utentes que garantem a sincronização entre si. O esquema geral da solução será então;

```
shared  int a, b, c[];    // vars partilhadas
main()
{
     // Se necessário, aguarda por um balcão livre, x
     balcao (x)  ; utente é atendido no balcao x
     // ...
}
```

Complete este programa, assumindo a existência de memória partilhada entre processos (declarada com `shared`) e operações `cria_semaforo`, `P` e `V`. Procure resolver a questão para um número $N > 5$ de balcões de atendimento. Em caso de dificuldade, resolva para $N = 1$[2].

2 Explique como, na questão anterior, se consegue partilhar um conjunto de endereços de memória por vários processos e simultaneamente garantir que cada processo mantém as suas próprias variáveis locais e globais.

II

Desenvolva um comando `atrasa` com um argumento numérico que, quando devidamente invocado (ex: `cat /etc/passwd | atrasa 10`) apresenta em STDOUT o input presente em STDIN, introduzindo no entanto um atraso temporal em segundos entre cada linha, consoante o argumento especificado. O envio dos sinais SIGUSR1 e SIGUSR2 a um processo `atrasa` deve permitir aumentar e diminuir o atraso em 1 segundo, respectivamente.

III

Tirando partido do comando "`sort`" (que nesta pergunta deve ser executado sem mais argumentos) que ordena o conteúdo recebido no STDIN e o emite no STDOUT, pretende-se criar a seguinte funcionalidade: Um utilizador que queira ordenar um ficheiro chamado "`nome.txt`" presente na directoria local, deve escrever o nome do mesmo para um pipe com o nome "`ordenar`". Tal pode ser feito via "`echo nome.txt > ordenar`". Faça um programa que leia repetidamente desse pipe e produza ficheiros ordenados com a extensão adicional "`sorted`". Neste exemplo seria gerado o ficheiro "`nome.txt.sorted`".

Protótipos de algumas funções e chamadas ao sistema relevantes

Processos

- `pid_t fork(void);`
- `void exit(int status);`
- `pid_t wait(int *status);`
- `WIFEXITED(status);`
- `WEXITSTATUS(status);`
- `int execlp(const char *file, const char *arg, ...);`
- `int execvp(const char *file, char *const argv[]);`

Sinais

- `void (*signal(int signum, void (*handler)(int)))(int);`
- `int kill(pid_t pid, int signum);`
- `int alarm(int seconds);`
- `int pause(void);`
- `SIGKILL, SIGINT, SIGALRM, SIGQUIT, SIGUSR1, SIGUSR2, SIGCHLD, ...`

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);`
- `int mkfifo(const char *path, mode_t mode);`

[1] Cotação — 7+6+7

[2] Nesse caso a cotação será consideravelmente inferior.