UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

U.C. 21111

Sistemas Operativos

27 de junho de 2016

INSTRUÇÕES

- Leia estas instruções na totalidade antes de iniciar a resolução do teste.
- O enunciado do teste é constituído por 2 grupos de questões, tem 3 páginas e termina com a palavra FIM.
- Se o seu exemplar não estiver completo ou nele se verificar qualquer outra deficiência, por favor dirija-se ao professor vigilante.
- O teste deve ser resolvido na sua totalidade em folhas de respostas.
- Nas respostas, tenha a preocupação de utilizar uma letra legível.
- Todas as respostas devem ser escritas unicamente com caneta azul ou preta.
- O teste é SEM CONSULTA. Todos os elementos necessários à resolução são fornecidos no enunciado.
- Não é permitido utilizar máquina de calcular.
- As cotações são indicadas por grupo e nas próprias questões.
- Nas questões de escrita de programas, a sua correção terá em conta critérios de proficiência e compreensibilidade do código (legibilidade, indentação, estrutura, comentários e explicação geral).
- O não cumprimento das instruções implica a anulação das respetivas questões.
- O tempo de realização do teste é de 150 minutos.

Grupo I [12 valores]

1.1. [1.2] Comente a seguinte frase: "O sistema operativo é um computador virtual".

1.2. [1.2] Caracterize resumidamente o conceito de sistema operativo de tempo real e as suas variantes.

1.3. [1.2] Explique o conceito de pseudoparalelismo.

1.4. [1.2] Comente possíveis razões para existirem tarefas.

1.5. [1.2] Explique em que consiste um semáforo e o seu modo de operação.

1.6. [1.2] Explique em que consiste o escalonamento de um processo e descreva quatro situações que possam levar o SO a efectuá-lo.

1.7. [1.2] Explique sucintamente em que consistem os problemas da Realocação e da Protecção quando se pretende que vários programas estejam simultâneamente em memória.

1.8. [1.2] Considere um sistema de memória virtual com um esquema de Tabela de Páginas mononível. Supondo que não ocorre uma falha de página, descreva qualitativamente como é obtido o endereço físico a partir do endereço virtual. Considere páginas de igual dimensão.

1.9. [1.2] Explique o significado de directoria de trabalho (ou corrente) e relacione-o com o conceito de "pathname"relativa.

1.10. [1.2] No âmbito do software de I/O, explique o conceito de "independência do dispositivo".

Grupo II [8 valores]

2.1. [3] Escreva um programa em linguagem C que crie um subprocesso e que com recurso a uma função exec() execute o comando "cp f1.txt f2.txt". O comando cp encontra-se na directoria /bin. O processo pai deve testar a ocorrência de erro na criação do processo filho assim como esperar que o processo filho termine.

2.2. [5] Escreva um programa multitarefa em linguagem C segundo a norma POSIX que transfira por blocos os elementos de um vector `int x[1000]` para um vector `int y[1000]`. Os vectores constituem variáveis globais ao programa e admite-se que foram devidamente inicializados.

A tarefa principal deve criar 10 subtarefas em que cada uma em paralelo (ou em pseudo-paralelismo) copia um bloco de 20 elementos do vector `x[]` para o vector `y[]`. O bloco origem em `x[]` tem início no elemento de menor índice ainda não transferido. O bloco destino em `y[]` tem início no elemento de menor índice ainda não preenchido. A tarefa principal deve esperar que todas as subtarefas terminem.

Nota 1: com a estratégia indicada, a ordem dos elementos em `y[]` será diferente (por blocos) da ordem em `x[]`, dependendo da ordem do escalonamento das tarefas.

Nota 2: planeie cuidadosamente como é dividido o trabalho entre as sub-tarefas e como é efectuada a sincronização e a comunicação da informação necessária à resolução do problema entre todas as tarefas.

Formulário

```
#include <stdlib.h>
int system(char *string);

#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
pid_t getpid(void);
pid_t getppid(void);

#include <unistd.h>
unsigned int sleep(unsigned int seconds);
extern char **environ;
int execl(char *path, char *arg, ...);
int execlp(char *file, char *arg, ...);
int execle(char *path, char *arg , ..., char *envp[]);
int execv(char *path, char *argv[]);
int execvp(char *file, char *argv[]);
int execve(char *path, char *argv [], char *envp[]);

#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *status);

#include <pthread.h>
int pthread_create(pthread_t *thread, pthread_attr_t *attr,
   void *(*start_routine)(void*), void *arg);
int pthread_attr_init(pthread_attr_t *attr);
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
#define PTHREAD_CREATE_DETACHED
#define PTHREAD_CREATE_JOINABLE
int pthread_join(pthread_t thread, void **value_ptr);
int pthread_mutex_init(pthread_mutex_t *mutex,
   pthread_mutexattr_t * attr);
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

FIM