UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

U.C. 21111

Sistemas Operativos

03 de julho de 2014

INSTRUÇÕES

Para a resolução do teste, leia as seguintes informações e instruções, antes de responder

- O enunciado do teste tem **4** páginas, sendo constituído por dois grupos de questões, I e II, com a cotação total de 20 valores.
- Nas respostas, tenha a preocupação de utilizar uma **letra legível** por outra pessoa.
- O grupo I é constituído por questões do tipo resposta aberta, com a cotação total de 12 valores.
- O grupo II é constituída por questões de escrita de Software, com a cotação total de 8 valores.
- As cotações são indicadas nas próprias questões/alíneas.
- Todas as respostas devem ser escritas unicamente com caneta azul ou preta.
- É permitido utilizar máquina de calcular.
- O não cumprimento das instruções implica a anulação das respectivas questões ou do teste.
- O tempo de realização do teste é de 150 minutos.
- Verifique se o teste está completo e termina na palavra FIM.

Grupo I

1.1. [1.2] Uma das principais funções de um SO é apresentar aos programas (aplicações) e programadores uma interface virtual baseada em abstrações, dita máquina estendida. Explique porquê.

1.2. [1.2] Relacione e caracterize os conceitos de multiprogramação e de partilha no tempo (timesharing).

1.3. [1.2] O que entende por uma hierarquia de processos ?

1.4. [1.2] Em que consiste uma tabela de processos ?

1.5. [1.2] Justifique porque não se deve programar com suposições baseadas no tempo de execução do código.

1.6. [1.2] Comente possíveis razões para existirem tarefas.

1.7. Considere um sistema com memória virtual e paginação. O espaço de endereçamento virtual é de 22 bits (4MB) e em determinado momento a tabela de páginas (mononível) tem o seguinte conteúdo,

nº entrada (decimal)	conteúdo (binário)	bit presente/ausente (1/0)
1023	00101	1
1022	00000	0
...	...	...
4	00000	0
3	11000	0
2	01010	1
1	10010	1
0	00000	1

1.7.1. [0,8] Considere as dimensões de página virtual e de página física (page frame) iguais. Indique justificando a dimensão da página utilizada e do espaço de endereçamento físico.

1.7.2. [0,8] Considere o endereço virtual 00 0000 0010 0010 1010 1101. Indique justificando o endereço físico correspondente.

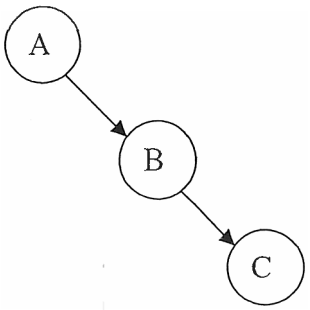
1.7.3. [0,8] Considere o endereço virtual 11 1111 1110 1001 1100 1001. Explique sucintamente o que acontece quando o programa efectua uma referência a este endereço.

1.8. [1.2] Qual é a principal desvantagem da implementação de um sistema de ficheiros baseado numa Tabela de Alocação de Ficheiros (FAT) ? Dê um exemplo numérico.

1.9. [1.2] Em que consiste a técnica de spooling ?

Grupo II

2.1. [3] Escreva um programa em linguagem C padrão que crie uma hierarquia de processos em árvore como mostra a figura. Cada processo deve imprimir na saída padrão uma mensagem do tipo "`Processo X`", onde X significa A, B ou C conforme o caso. Cada processo deve esperar que os seus processos filhos (directos) terminem, antes dele próprio terminar. O programa não necessita de testar erros nas chamadas às funções.



2.2.1. [4.5] Escreva um programa multitarefa em linguagem C segundo a norma POSIX que analise se os elementos de um vector `int x[]` de dimensão `int nx` são números pares ou ímpares e os transfira para um vetor `y[]`, colocando os elementos pares a partir da base (`y[0]`) e os ímpares a partir do topo (`y[nx-1]`). O vector `x[]` e a sua dimensão constituem variáveis globais ao programa e admite-se que foram devidamente inicializadas.

A tarefa principal deve criar dez subtarefas em que cada uma em paralelo (ou em pseudo-paralelismo) remove o último elemento do vector `x[]` e o transfere para o vetor `y[]`, após o que deve dar oportunidade às outras tarefas de também realizarem trabalho. Note que a dimensão `nx` do vector `x[]` diminui de uma unidade cada vez que uma tarefa opera no vector, devendo as tarefas terminarem quando se atingir `nx=0`, caso em que `y[]` contém todos os elementos de `x[]`. A tarefa principal deve esperar que todas as subtarefas terminem antes de terminar.

Nota: 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 as onze tarefas.

2.2.2. [0.5] Considere que o programa da alínea anterior está num ficheiro de nome `parx.c`. Indique o comando Linux necessário à sua compilação e criação de um ficheiro executável.

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