

**U.C. 21111**

**Sistemas Operativos**

**30 de setembro de 2019**

### **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] Explique em que consiste um interpretador de comandos (shell) e o seu modo de operação. Este é considerado como pertencente ao sistema operativo ?
- 1.3. [1.2] Relacione os conceitos de troca de contexto (context switch) e de multi-programação.
- 1.4. [1.2] Comente possíveis razões para existirem tarefas.
- 1.5. [1.2] O que entende por uma região crítica?
- 1.6. [1.2] Explique em que consiste um semáforo e o seu modo de operação.
- 1.7. [1.2] O espaço de endereçamento de um processo é dividido em três partes (segmentos). Caracterize-os.
- 1.8. [1.2] Explique em que consiste uma Tabela de Tradução de Endereços (TLB) e a importância da sua existência.
- 1.9. [1.2] O que é e que funções desempenha o Master Boot Record (MBR) ?
- 1.10. [1.2] Qual o nome do circuito de controlo que permite a comunicação directa entre um dispositivo de I/O e a memória RAM ?

### Grupo II [8 valores]

- 2.1. [3] Escreva um programa em linguagem C que crie um sub-processo e que com recurso à função `execv()` execute um comando que produza os mesmos resultados que a execução de `"ls -l /home > aaa"` na linha de comandos. O comando `ls` encontra-se na directoria `/bin`. O processo pai deve esperar que o processo filho termine.
- 2.2. [5] Escreva um programa multitarefa em linguagem C segundo a norma POSIX que determine a soma  $S$  dos elementos de um vector `int x[]` de dimensão `int nx`. O vector e a sua dimensão constituem variáveis globais ao programa e admite-se que foram devidamente inicializados.

A tarefa principal deve criar dez subtarefas em que cada uma em paralelo (ou em pseudo-parallelismo) substitui de cada vez dois elementos do vector `x[]` por um único correspondente à sua soma, 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 terminar quando se atingir `nx=1`, caso em que `x[0]` contém a soma total do vector. A tarefa principal deve esperar que todas as subtarefas terminem e depois deve imprimir o resultado final e 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.

## 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 execl_e(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