Nome: Data: 2017/11/04 Curso: Engª Informática Nº

Programação Avançada / 1-a T (C)
1º Semestre 2017/18

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
Programação Avançada
2º ano - Engenharia Informática
Ano lectivo 2017/18 1ºSemestre

Prova Escrita (sem consulta) – Enunciado C

2017-11-04 **Duração:** 50 min

Nome Completo:_____ N.º aluno:_____

Todas as perguntas devem ser resolvidas no enunciado!

[20 valores]

- Considere que todas as perguntas deste grupo são independentes.
- **Todas as perguntas respondidas incorretamente ou de forma ambígua descontam 25% da cotação da pergunta.**
- Selecione a resposta mais completa para cada uma das seguintes questões.
- Prova sem consulta.
- É expressamente proibído o uso de telemóveis ou de qualquer outro dispositivo eletrónico.

Tabela de respostas

Escreva, de forma **legível**, no retângulo reservado para o efeito, a **letra** da opção que considera a resposta certa. Caso não pretenda responder à pergunta, escreva "X" no meio do retângulo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Estudante																				
Professor																				

1. **[1 valor] Uma chamada ao sistema …**
   a) é executada em modo *kernel / sistema*
   b) é uma forma de acesso aos serviços do sistema operativo
   c) requer que o sistema comute para modo dito privilegiado ou sistema
   d) todas as anteriores

2. **[1 valor] A falta de sincronização apropriada numa aplicação com múltiplas threads pode levar a...**
   a) competição por recursos (*race condition*)
   b) comportamento não determinístico da aplicação
   c) ocorrência de falhas do tipo *heisenbugs*
   d) todas as anteriores

3. **[1 valor] Para se obter a listagem das chamadas ao sistema efetuadas por uma dada aplicação…**
   a) usa-se o comando `ps`
   b) usa-se o comando `ltrace`
   c) usa-se o comando `strace`
   d) nenhuma das anteriores

4. **[1 valor] Na aplicação bash, o identificador $$ corresponde ...**
   a) ao PID do processo que executa a aplicação `bash` empregue pelo utilizador
   b) ao nome de *login* do utilizador que está a fazer uso da aplicação `bash`
   c) ao UID do utilizador que executa a aplicação `bash` empregue pelo utilizador
   d) nenhuma das anteriores

5. **[1 valor] Considera o seguinte código fonte que compila sem avisos nem erros num sistema Linux.**

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(void) {
    fork();
    for(int i=0;i<4;i++){
           if( fork() == 0 ){
                  printf("Processo PID=%d\n", getpid());
                  exit(0);
           }
    }
    return 0;
}
```

Caso a execução decorra de forma normal (sem erros), quantas linhas são escritas para a saída padrão?

a) 8
b) 16
c) 32
d) 4

6. **[1 valor] A função `execlp` (protótipo: `int execlp(const char *file, const char *arg, ...);`)...**
   a) quando é executada com sucesso leva a que a imagem do processo chamante seja substituída pelo conteúdo do ficheiro indicado pelo parâmetro `file`
   b) tem que receber a string “END” como indicador do último parâmetro para funcionar convenientemente
   c) apenas pode ser executada num processo corrido em modo administrador (*root*)
   d) nenhuma das anteriores

7. **[1 valor] A função pthread_create tem o seguinte protótipo:**

```
int pthread_create(pthread_t *thread,
        const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);
```

O <u>terceiro</u> parâmetro da função `pthread_create` corresponde…

a) a um ponteiro para função do tipo `void Func(void);`
b) a um ponteiro para função do tipo `void *Func(void);`
c) a um ponteiro para função do tipo `void Func(void*);`
d) Nenhuma das anteriores

8. **[1 valor] Na norma pthread, a função** `pthread_mutex_trylock` **(protótipo:**
`int pthread_mutex_trylock(pthread_mutex_t *mut);`**)...**

   a) solicita uma operação de *lock* ao *mutex* `mut`, bloqueando a *thread* chamante caso o *mutex* `mut` esteja na posse de outra *thread*

   b) valida se o *mutex* `mut` é um mutex válido

   c) solicita uma operação de lock ao *mutex* `mut`, retornando imediamente com indicação de erro caso o *mutex* `mut` esteja na posse de outra thread

   d) nenhuma das anteriores

9. [1 valor] A diretiva `#pragma omp`..

   a) é uma diretiva do préprocessador da linguagem C

   b) deve ser ignorada pelos préprocessadores de linguagem C que não suportem a norma OpenMP

   c) é empregue pelo programador para marcar um bloco de código a ser executado de forma concorrente pela norma OpenMP

   d) todas as anteriores

10. [1 valor] Na norma pthread, uma variável de condição…

   a) apenas deve ser acedida em exclusão mútua quando se faz uso das funções `pthread_cond_wait` e `pthread_cond_timedwait`

   b) pode ser manipulada através das funções `pthread_cond_wait`, `pthread_cond_timedwait`, `pthread_cond_signal` e `pthread_cond_broadcast`

   c) é empregue para garantir a exclusão mútua no acesso a uma variável partilhada

   d) nenhuma das anteriores

11. [1 valor] Num núcleo preemptivo de um sistema operativo...

   a) apenas os processos de entrada e saída têm acesso ao CPU

   b) um processo ou *thread* em execução em modo *kernel* pode ser interrompido

   c) um processo em execução apenas pode ser interrompido quando está a executar em modo utilizador

   d) nenhuma das anteriores

12. [1 valor] Um ficheiro binário executável é...

   a) Um ficheiro que contém instruções que se destinam a ser executadas pelo processador

   b) Um ficheiro com um formato bem definido que é executado no contexto do sistema operativo sob a forma de processo

   c) Um ficheiro que resulta usualmente da compilação de código fonte para código executável

   d) Todas as anteriores

13. [1 valor] O conjunto de bits `00111100.10100001` quando interpretado como uma variável inteira sem sinal de 16 bits corresponde a …

a) `72345` em base `10`

b) `0x3CA1` em base hexadecimal

c) `07291` em base octal

d) Nenhuma das anteriores

14. [1 valor] Considere o código em linguagem C da função func**:**

```
double func(double radius){
      static double b = 3.14 * radius * 2.0;
      return b;
}
```

A função func é…

a) reentrante

b) não reentrante

c) recursiva

d) nenhuma das anteriores

15. [1 valor] Considere que A e B são duas threads de um mesmo processo. Considere ainda que M1 e M2 são dois mutexes e que lock(M1) significa operação de lock no mutex M1 e unlock(M1) representa operação de unlock sobre o mesmo mutex M1.

thread A	thread B
lock(M1)	lock(M2)
lock(M2)	lock(M1)
(…)	(…)
unlock(M2)	unlock(M1)
unlock(M1)	unlock(M2)

Considerando somente a sequência de operações descrita na tabela e que não ocorre nenhuma terminação inesperada de threads (i.e., não há "crashes"), poderá a sequência de operações originar um *deadlock*?

a) Sim, dado que as threads operam os mutexes por ordem diferente: a thread A efetua a sequência M1/M2 enquanto que a thread B efetua a sequência M2/M1

b) Não, pois ambas as threads estão a usar os mesmos mutexes

c) Sim, porque ambas as threads estão a libertar (unlock) os mutexes pela ordem inversa de que foram tomados (lock)

d) Nenhuma das anteriores

16. [1 valor] Na programação, o operador >>>…

a) está disponível na linguagem python

b) efetua rotação à direita com extensão de sinal

c) permite multiplicar por 2

d) nenhuma das anteriores

17. [1 valor] Considere o seguinte código em linguagem C que compila sem avisos e sem erros com o compilador gcc.

```
#include <stdio.h>
int main(void){
	char value = 0;
	value--;
	if( value < 0 ){
		printf("<0: %d\n", value);
	}else{
		printf(">0: %u\n", value);
	}
	return 0;
}
```

Quando o programa é executado na máquina virtual Linux de Programação Avançada, produz a seguinte saída:

```
<0: -1
```

Isso significa que…

a) o compilador implementa o tipo `char` como sendo um `char` com sinal (`signed char`)

b) o compilador implementa o tipo `char` como sendo um `char` sem sinal (`unsigned char`)

c) não é possível tirar conclusões sobre a existência ou não de sinal no tipo `char`

d) nenhuma das anteriores

18. [1 valor] Considere o seguinte código C:

```
uint16_t C = ((1 << 1) | (1 << 3) | (1 << 10)) & (0xFFFF);
```

O que se pode dizer sobre o valor da variável C **<u>após</u>** a execução do código?

a) A variável C fica com o valor `77324 (base 10)`

b) A viariável C fica com o valor `02012 (base 8)`

c) A variável C fica com o valor `0x40b (base 16)`

d) Nenhuma das anteriores

19. [1 valor] Na norma *pthread*...

a) Um *mutex* deve ser empregue para garantir a exclusão mútua

b) Uma variável de condição deve ser empregue para assinalar eventos entre *threads*

c) Uma variável de condição deve ser empregue juntamente com um *mutex*

d) Todas as anteriores

20. [1 valor] O comando `ulimit`…

a) É um comando interno da `bash`

b) Permite configurar limites para certo tipos de recursos, como por exemplo, o tamanho de um *core file*

c) A documentação para o `ulimit` está acessível através de `help ulimit`

d) Todas as anteriores