

Prova Escrita (sem consulta) – Exame Normal - Enunciado D

2018-02-03

Duração: 50 min

Nome completo: _____ N.º aluno: _____

Regime: Diurno [] Pós-laboral []

Todas as perguntas devem ser resolvidas no enunciado!


Grupo I [17 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 proibido 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 | |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----------|
| Estudante | | | | | | | | | | | | | | | | | | Grupo II |
| Professor | | | | | | | | | | | | | | | | | | |

- [1 valor] Numa máquina com arquitetura de ordenação de bytes *little endian*, a codificação UTF-8 do emoji  é 0xEE8487. Como será essa mesma codificação numa máquina com arquitetura *big endian*?
 - 0x8784EE
 - 0xEE8487
 - 0x8487EE
 - Nenhuma das anteriores
- [1 valor] Considere o seguinte número expresso em formato hexadecimal: 0x22446688:
 - Num sistema com representação *big endian*, o primeiro byte a ocorrer em memória é 0x88
 - Num sistema com representação *little endian*, o primeiro byte a ocorrer em memória é 0x88
 - Caso seja interpretado como um inteiro de 32 bits, o inteiro representado pela sequência de bits é negativo
 - Nenhuma das anteriores

3. [1 valor] Considere um inteiro de 24 bits com sinal. O menor e maior valor suportados são respetivamente...

- a) -2^{24} e $+2^{24}$
- b) -2^{23} e $+2^{23}$
- c) -2^{23} e $+2^{23}-1$
- d) $-2^{23}-1$ e $+2^{23}$

4. [1 valor] O endereço representado por ::1::

- a) Corresponde ao endereço IPv6 0:0:0:0:1:0:0:0
- b) Corresponde ao endereço IPv4 127.0.0.1
- c) Não é um endereço válido em IPv6
- d) Nenhuma das anteriores

5. [1 valor] Em que situação pode uma chamada à função `bind` falhar, considerando que se está a fazer uso do protocolo TCP? Protótipo: `int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);`

- a) O parâmetro `sockfd` passado à função é inválido
- b) O porto solicitado para efetuar o `bind` já se encontra ocupado
- c) O porto solicitado é inferior a 1024 e o processo chamante não tem permissões de *root*
- d) Todas as anteriores

6. [1 valor] O valor devolvido pela função `recv` (protótipo: `ssize_t recv(int sockfd, void *buf, size_t len, int flags);`) corresponde...

- a) A 0 (zero) caso a outra extremidade comunicante tenha fechado a ligação
- b) A -1 caso tenha ocorrido um erro na chamada à função
- c) Ao número de octetos (bytes) recebidos, caso a chamada seja bem sucedida
- d) Todas as anteriores

7. [1 valor] A função `shutdown` da API `socket` (protótipo: `int shutdown(int sockfd, int how);`)...

- a) possibilita o encerramento parcial de uma ligação TCP
- b) possibilita o encerramento parcial de uma ligação UDP
- c) possibilita o encerramento parcial de uma ligação ICMP
- d) nenhuma das anteriores

8. [1 valor] A execução de um determinado comando produziu a saída que é parcialmente mostrada na listagem seguinte:

```
(...)  
strlen("00:00:00")           = 8  
fwrite(" 00:00:00", 12, 1, 0xb76f5d60) = 1  
strlen("ps")                 = 2  
(...)
```

Trata-se do comando...

- a) `ls /tmp`
- b) `ps aux`
- c) `ltrace ps`
- d) nenhuma das anteriores

9. [1 valor] Num sistema Linux, foi executada a seguinte linha de comando:

```
cat /proc/sys/kernel/pid_max
```

produzindo o resultado **32768**. Isso significa que...

- a) quando é atribuído o PID 32768 o sistema Linux é forçado a fazer um *reboot* por forma a renovar a atribuição de PID
- b) é possível executar até 32768 processos simultaneamente no sistema
- c) o número máximo de utilizadores que podem estar registados no sistema é 32768
- d) nenhuma das anteriores

10. [1 valor] Considere o pseudo-código da função `func`. A linha `lock(&mutex_A)` corresponde à operação de **lock** no mutex ***mutex_A*** e `unlock(&mutex_A)` à operação de ***unlock***.

```
double func(double radius, pthread_mutex_t mutex_A){
    static double b;
    lock(&mutex_A);
    b = 3.14 * radius * 2.0;
    unlock(&mutex_A);
    return b;
}
```

A função `func` é...

- a) recursiva
- b) reentrante
- c) *thread-safe*
- d) nenhuma das anteriores

11. [1 valor] Considere o seguinte código em linguagem C:

```
uint32_t A = 0x0000;
uint32_t B = A ^ (~A);
```

O que se pode dizer sobre o valor da variável **B** após a execução do código?

- a) A variável B fica com o valor 0xFFFF
- b) A variável B fica com o valor 0xFFFFFFFF
- c) A variável B fica com o valor 0x0
- d) Nenhuma das anteriores

12. [1 valor] Considerando um sistema Linux a correr num processador do tipo X86_64. Qual das seguintes variáveis é afetada pela *endianess* do sistema?

- a) `int a = 0x1122;`
- b) `int16_t b = 0x12;`
- c) `uint64_t c = 0x21;`
- d) Todas as anteriores

13. [1 valor] Exemplos de protocolos aplicacionais que fazem uso do protocolo UDP são...

- a) Todas as seguintes
- b) DNS
- c) NTP
- d) TFTP

14. [1 valor] No protocolo HTTP, os códigos que são empregues pelo servidor para reportarem erros existentes num pedido do cliente HTTP são...

- a) Classe 1xx
- b) Classe 2xx
- c) Classe 4xx
- d) Classe 5xx

15. [1 valor] No Unix, a função `int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);` serve para...
- a) Instalar uma função para tratamento do *signal* especificado pelo parâmetro `signum`
 - b) Reagir imediatamente quando é recebido o *signal* `signum`, encerrando o processo chamante
 - c) Desativar o comportamento por omissão do *signal* `SIGSTOP` ou do *signal* `SIGKILL`
 - d) Todas as anteriores
16. [1 valor] Um ambiente *multithread*, isto é, onde um processo pode ter múltiplas *threads*, é benéfico porque...
- a) permite a abstração de tarefas, possibilitando a divisão da aplicação pelas suas várias tarefas
 - b) possibilita a expressão do paralelismo da aplicação com a distribuição do processamento pelos vários cores do sistema
 - c) a mudança de contexto de *threads* é substancialmente menos onerosa do que as mudanças de contexto de processos
 - d) todas as anteriores
17. [1 valor] Um ficheiro `.h` deve conter...
- a) Protótipos de funções públicas
 - b) Protótipos de variáveis globais “públicas”
 - c) Um `#ifdef` e respetivo `#endif` para evitar múltiplas inclusões do ficheiro `.h`
 - d) Todas as anteriores

Grupo II [3 valores]

Deve escrever com caligrafia legível. Respostas ilegíveis não são consideradas, resultando na nota 0 (zero valores) à alínea em apreço.

18. [3 valores] Como sabe, a função `recv` da API *socket* tem o seguinte protótipo:

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

Considere o seguinte bloco de código em linguagem C

```
1. int sock = socket(AF_INET, SOCK_STREAM, 0);
2. (...)
3. char str[256];
4. int flags = 0;
5. ssize_t ret_recv = recv(&sock, &str, strlen(str), flags);
6. (...)
```

Existem pelo menos três erros na chamada à função `recv` (linha 5). Identifique cada um dos três erros, indicando ainda a respetiva correção. Considere que o *socket* ‘sock’ é convenientemente criado e se encontra operacional.

Erro 1: _____

Correção Erro 1: _____

Erro 2: _____

Correção Erro 2: _____

Erro 3: _____

Correção Erro 3: _____
