ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Licenciatura em Eng. Informática
PRCMP – Princípios da Computação

Exame da Época Normal
19 de janeiro de 2024

Duração: 60 minutos. Sem consulta. Responda no enunciado.

Número: _____ **Nome:** _____

Seleccione a resposta correcta para cada uma das questões. (Resposta errada: 0 valores.)

1. Considere a representação do número negativo -2 num computador.
   (a) Pode ser representado como número inteiro sem sinal.
   (b) Pode ser representado em complemento para 2, na palavra de 8 bits: `00000010`.
   (c) Pode ser representado na palavra de 8 bits com sinal: `-00000010`.
   (d) Pode ser representado em vírgula flutuante pela norma IEEE 754.
2. Admita um programa escrito numa linguagem de programação de alto-nível.
   (a) O código-fonte desse programa é directamente executado pelo processador.
   (b) Um compilador traduz e executa o programa linha por linha.
   (c) O programa executará mais depressa se compilado em vez de interpretado.
   (d) Um interpretador traduz o programa para código-máquina e liga-o a código disponibilizado por bibliotecas, produzindo um ficheiro executável.
3. De acordo com a arquitetura de von Neumann...
   (a) um computador necessita de um sistema operativo para operar eficientemente os seus recursos.
   (b) as instruções e os dados de um programa em execução são armazenados em memória, embora em regiões distintas.
   (c) um programa em execução tem as instruções num ficheiro e os dados em memória.
   (d) não é possível a execução de múltiplos programas simultaneamente.
4. Que função não faz parte das responsabilidades de um processador (CPU)?
   (a) Armazenar os dados de um programa.
   (b) Realizar operações aritméticas e lógicas.
   (c) Transferir dados com a memória.
   (d) Descodificar os códigos de operações (*opcodes*).
5. Qual a função principal da memória secundária num computador?
   (a) Armazenar dados (e.g. variáveis) para a execução imediata de programas.
   (b) Permitir a comunicação entre diferentes CPU no mesmo computador.
   (c) Fornecer armazenamento de longo prazo para dados e programas, mesmo quando o computador está desligado.
   (d) Armazenar temporariamente na CPU os dados usados durante a execução de uma instrução.

6. Qual dos seguintes aspectos não é geralmente especificado pela arquitectura do conjunto de instruções (*Instruction Set Architecture*, ISA)?
   (a) O conjunto de instruções suportado pela CPU.
   (b) O tamanho e a organização dos registos.
   (c) O formato dos códigos de operações (*opcodes*).
   (d) O número de núcleos presentes no processador.
7. É necessário colocar a zero os 16 bits mais significativos de uma palavra de 32 bits. Uma solução é aplicar à palavra...
   (a) 16 deslocamentos lógicos para a direita seguidos de 16 deslocamentos lógicos para a esquerda.
   (b) a operação `AND` com a máscara `0x0000FFFF`.
   (c) a operação `OR` com a máscara `0x0000FFFF`.
   (d) a operação `XOR` com a máscara `0x0000FFFF`.
8. Num sistema multiprogramado, a protecção de memória...
   (a) impede que um processo modifique a sua pilha.
   (b) impede a partilha de bibliotecas ligadas dinamicamente pelos vários processos.
   (c) impede que um processo modifique a sua secção de texto (ou código).
   (d) isola fisicamente cada processo atribuindo-o uma CPU dedicada.
9. A técnica de *swapping* permite...
   (a) ter mais do que um processo activo no computador.
   (b) usar mais memória do que a memória física do computador.
   (c) trocar o processo em execução pelo processador.
   (d) executar processos que se encontram em memória secundária.
10. Quais são os principais objetivos de uma interface de utilizador num sistema operativo?
    (a) Proporcionar um ambiente de utilização visualmente apelativo.
    (b) Garantir a segurança total do sistema.
    (c) Minimizar o consumo de recursos do sistema.
    (d) Facilitar ao utilizador o acesso às funcionalidades do sistema operativo.
11. Um programa tem que realizar uma operação num dispositivo.
    (a) O programa controla directamente o dispositivo, após obter a sua configuração do sistema operativo.
    (b) O programa controla directamente o dispositivo, após obter autorização do sistema operativo.
    (c) O programa controla directamente o dispositivo, se o computador não tiver sistema operativo.
    (d) O programa controla o dispositivo colocando a operação a realizar num endereço específico da memória.
12. Os computadores que controlam sistemas físicos (e.g. automóveis, robots) têm requisitos não-funcionais específicos.
    (a) Têm de ser bastante responsivos na interacção com o utilizador.
    (b) Têm de realizar o maior número de cálculos por unidade de tempo.
    (c) Têm de conseguir de lidar com um aumento no número de tarefas sem comprometer o desempenho geral.
    (d) Têm de determinar resultados correctos dentro de prazos temporais estritos.

13. Em que consiste o conceito de sistemas de partilha de tempo (*time-sharing*)?

    (a) "Partilha de tempo" refere-se à atribuição de recursos a cada utilizador, por intervalos de tempo fixos. Este modelo permite que utilizadores prioritários tenham sempre acesso preferencial aos recursos do sistema.

    (b) "Partilha de tempo" envolve a execução de programas em sequência para garantir o desempenho máximo de cada um. Um benefício significativo é a simplicidade da gestão do sistema.

    (c) "Partilha de tempo" refere-se à prática de compartilhar recursos de computação (CPU, memória, etc.) simultaneamente entre vários utilizadores. Este modelo permite que vários utilizadores interajam com o sistema ao mesmo tempo, aumentando a eficiência do sistema.

    (d) "Partilha de tempo" refere-se à execução simultânea de várias instâncias idênticas de um programa. A redundância assim obtida confere maior fiabilidade em caso de falhas.

14. Num sistema multiprogramado, o escalonamento de processos pode beneficiar do mecanismo de preempção. Em que situações uma preempção pode ocorrer?

    (a) A preempção ocorre quando um processo atinge seu tempo máximo de execução permitido e é interrompido pela CPU para permitir que outro processo execute, garantindo que nenhum processo monopolize a CPU indefinidamente.

    (b) A preempção ocorre quando um processo cede voluntariamente a CPU após a conclusão da sua execução, garantindo uma transição suave entre processos.

    (c) A preempção ocorre quando um processo cede voluntariamente a CPU durante uma secção de código computacionalmente intensiva, evitando o aquecimento da CPU.

    (d) A preempção ocorre quando um processo voluntariamente cede a CPU após atingir o limite de tempo máximo de execução, proporcionando uma execução justa ao conjunto de processos.

15. Em Unix, a *shell* é uma interface da linha de comando que permite ao utilizador carregar e executar programas de uma forma simples. O Unix traz um conjunto de programas utilitários que realizam tarefas específicas muito frequentes.

    (a) O comando `mv` permite alterar o nome de um ficheiro.

    (b) O comando `chmod` é utilizado para compactar ficheiros e directórios num formato específico.

    (c) O comando `cp` é usado para mover ficheiros ou diretórios de uma localização para outra, eliminando o original.

    (d) O comando `grep` é utilizado para procurar ficheiros que correspondam a um critério baseado nas propriedades dos ficheiros.

16. Num sistema com escalonamento preemptivo (*preemptive scheduling*)...

    (a) um processo no estado *Waiting* pode passar para o estado *Running*.

    (b) um processo no estado *Ready* pode passar para o estado *Waiting*.

    (c) um processo no estado *Running* pode passar para o estado *Ready*.

    (d) um processo no estado *Running* pode monopolizar o uso do processador, se não o ceder voluntariamente a outros processos.

Desenvolva um shell script que resolva o seguinte problema.

17. Um programador que participa em vários projectos tem necessidade de manter os seus repositórios locais frequentemente actualizados. Diariamente realiza uma tarefa fastidiosa: para cada repositório (1) entra no respectivo directório de trabalho e (2) executa o comando "`git pull`".

    Esta tarefa pode ser facilmente automatizável por um *shell script*; por sua vez, este *shell script* pode ser facilmente configurável por um ficheiro de texto que contenha os directórios de trabalho de cada projecto.

    Assuma que existe um ficheiro de texto com o caminho para um directório de trabalho por linha, como no exemplo seguinte:

```
/home/joaosilva/projects/ProjectA/
/home/joaosilva/projects/ProjectB/
/home/joaosilva/projects/TopSecretProject/
```

Escreva um *script* que cumpra com os seguintes requisitos:

R1: O utilizador indica o nome do ficheiro com os directórios de repositórios na linha de comandos.

R2: A falha de acesso a um ficheiro com os directórios de repositórios é irrecuperável.

R3: O comando "`git pull`" deve ser executado para cada directório de repositório válido.

R4: Um directório de repositório inválido deverá originar uma mensagem de erro.

SUGESTÃO: Um ficheiro pode ser lido linha-a-linha para uma variável do seguinte modo:

```
while read -r linha do
    echo $linha
done < nome_do_ficheiro
```

FIM