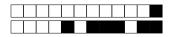
i.	1.	/	1	/	0	\sim	-
т	т.	/	1	/	O	u	┰

Eng. Informática/Ciências da Computação	Número:		
Teste Modelo de Sistemas Opera			
16 de Abril de 2025 – Duração:			
Instruções: Preencha o nome e o número de alimente, pinte completamente (mente, pinte completamente (messa sa caixa mais para assinalar as respostas que considera correctas Não use as áreas sombreadas! Importante: Este documento contém vários teste/exame. Note que o número e tipo de questõe (p.ex., será ajustado ao tempo para efetuar o messa Nos exemplos de código-fonte assuma que as corretamente e sem erros (a não ser que a contrário)	2 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 6 9 </th		
Processos • pid_t fork(void); • void _exit(int status); • pid_t wait(int *status); • pid_t waitpid(pid_t pid, int *status, int options); • WIFEXITED(status); • WEXITSTATUS(status); • int execlp(const char *file, const char *arg,); • int execvp(const char *file, char *const argv[]);	Sistema de Ficheiros int open(const char *pathname, int close(int fd); int read(int fd, void *buf, size_t int write(int fd, const void *buf, long lseek(int fd, long offset, int int pipe(int filedes[2]); int mkfifo(const char *path, mod int dup(int oldfd);	count); size_t count); whence);	
Grupo I	• int dup2(int oldfd, int newfd);	10 valores	
emplo de questões sobre a componente teórica			
nestão 1 — O escalonador de processos de um sistem processos intensivos em termos de CPU e I/O protecupação por parte do escalonador de processos?		_	
stifique a sua resposta.		5 .6 .7 .8 .9 1	
lestão 2 Considere um serviço que permite a ex e requerem interação frequente com os utilizadores. los os programas simultaneamente, é necessário escal	Como este serviço nem sempre tem ca		
sumindo que este serviço suporta mecanismos de des nte a sua execução e resumir a mesma mais tarde)			

Justifique a sua resposta indicando uma vantagem e uma possível preocupação ou desvantagem da sua esco-

0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1

lha.



Questão 3 Os sistemas operativos modernos utilizam paginação para gerir a memória dos processos em espaço do utilizador. Ainda, a paginação é muitas vezes combinada com mecanismos de *swapping* de páginas em memória para disco.

- 1. Indique **uma vantagem** de combinar paginação com mecanismos de *swapping*. Indique também **uma preocupação** que o sistema operativo deve ter para tornar o *swapping* de páginas eficiente. Justifique a sua resposta.
- 2. Ao configurar a área de swap num dado computador é preferível utilizar um sistema de ficheiros, ou utilizar diretamente uma partição (raw) de disco, para suportar a mesma? Justifique a sua resposta.

0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1

Questão 4 — Ao analisar o comportamento de acesso ao disco rígido (HDD) de um servidor, observou que algumas aplicações exibem mau desempenho (*i.e.*, os pedidos ao disco destas aplicações demoram muito tempo a serem servidos). Ao analisar o sistema operativo repara que este está configurado com um algoritmo de escalonamento Shortest Seek Time First (SSTF). Relembre que este algoritmo escolhe os próximos pedidos a servir de acordo com a proximidade dos cilindros do disco a que estes acedem (*i.e.*, minimiza o movimento da cabeça do disco).

Indique uma possível razão para o mau desempenho das aplicações. Ainda, indique um algoritmo de escalonamento alternativo que podia ser mais justo e melhorar o desempenho destas aplicações.

Questão 5 Imagine que pretendia desenvolver um serviço de orquestração (*i.e.*, execução e escalonamento) de tarefas (programas) num computador. Este serviço sabe, à partida, qual o tempo de execução de cada tarefa mas não possui quaisquer mecanismos de desafetação forçada de processos (*i.e.*, não consegue interromper um processo durante a execução de uma tarefa). Tendo em conta as considerações anteriores, assinale os algoritmos de escalonamento que poderia usar para o serviço em questão.

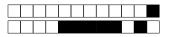
- A Multi-Level Feedback Queue (MLFQ).
- B Shortest Job First (SJF).
- C First-Come First-Served (FCFS).
- D Round-Robin (RR).
- E Nenhuma das respostas apresentadas está correta.

Questão 6 As técnicas de *paginação* e *segmentação* pretendem resolver problemas distintos relativamente à alocação de memória central pelos processos a correr num dado sistema operativo. A fragmentação interna e externa da memória central são dois fenómenos comuns que podem levar a desperdício na utilização da mesma. Assumindo que a técnica de paginação é configurada com um tamanho de página e *frame* idêntico, assinale as respostas verdadeiras.

- A segmentação e paginação não sofrem de qualquer tipo de fragmentação.
- B A segmentação de memória pode sofrer de fragmentação externa e interna.
- C A paginação de memória pode gerar fragmentação externa mas não interna.
- D A paginação de memória pode gerar fragmentação interna mas não externa.
- |E| Nenhuma das respostas apresentadas está correta.

Questão 7 Um sistema de ficheiros é responsável por decidir qual a melhor forma de armazenar os dados de ficheiros em disco. Considerando um disco rígido mecânico (HDD), é vantajoso armazenar os dados de cada ficheiro de forma contígua no disco. Concorda com esta afirmação? Assinale as respostas corretas

- A Não, acessos sequenciais ou aleatórios num disco HDD têm o mesmo desempenho.
- B Sim, tempos de acesso aleatório são mais altos do que tempos de acesso sequenciais em discos HDD.
- C Sim, já que tende a aumentar o tempo médio de seek do disco HDD.
- D Sim, já que tende a diminuir o tempo médio de seek do disco HDD.
- |E| Nenhuma das respostas apresentadas está correta.



Grupo II 10 valores

Exemplo de questões sobre a componente prática

Questão 8 Considere uma empresa que utiliza um único ficheiro escrito em formato binário para armazenar todos os registos individuais dos seus funcionários. Cada registo contém o nome, cargo e salário na empresa de um funcionário, de acordo com a seguinte estrutura:

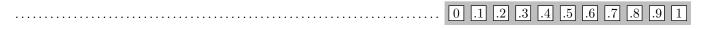
```
struct RegistoF {
char nome[20];
char cargo[20];
int salário;
};
```

- 1. Escreva a função void aumentaSalarios(char* ficheiro, char* cargo, int valor, long N, int P) que atualiza o *ficheiro* com N registos de forma aumentar em valor o salário dos funcionários com um dado cargo. A função deve desencadear uma atualização concorrente com P processos.
- 2. Escreva a função int validaSalarios(char* ficheiro, char* cargo) que valida o salário de todos os funcionários com um dado *cargo*. A função deve retornar 1 caso algum funcionário receba menos do que o salário mínimo. Caso contrário, deve retornar 0. Para resolver este exercício, deve tirar proveito dos seguintes **ficheiros** executáveis:
 - ./filtraCargo <ficheiro> <cargo> procura no ficheiro de registos os funcionários de um dado cargo e escreve para o stdout os respetivos registos.
 - ./validaMin lê registos do stdin, no mesmo formato produzido pelo ficheiro executável filtraCargo, e termina com código de saída 1, caso algum funcionário receba menos que o salário mínimo, ou 0 caso contrário.

Questão 9 Considere um programa tradutor que lê linhas de texto do stdin, traduz as mesmas para a língua portuguesa, e escreve as linhas traduzidas para o stdout. Este programa não recebe quaisquer argumentos.

- 1. Implemente a **função** void traduz_e_filtra(char* caminho_ficheiro, char* palavra_chave) que deve traduzir o conteúdo de um dado ficheiro, cujo caminho é passado como argumento (caminho_ficheiro), escrevendo para o **sdtout** apenas as linhas traduzidas que contêm uma certa palavra chave (palavra_chave).
 - Deve tirar proveito do programa **tradutor** e do comando **grep** na implementação desta função. Relembre-se que ao invocar o comando "**grep** SO", este irá ler linhas do **stdin** e apenas escrever para o **stdout** as linhas que contêm a palavra "SO".
- 2. Implemente a **função** void filtraN(char* caminhos_ficheiros[], int total_ficheiros, char* palavra_chave) que deve desencadear o processamento (*i.e.*, tradução e filtragem) de vários ficheiros em <u>paralelo</u>. Esta função recebe como argumentos uma lista de caminhos de ficheiros (*caminhos_ficheiros*), o número total de ficheiros (*total_ficheiros*), e uma palavra chave (*palavra_chave*). A sua solução deve utilizar a função traduz_e_filtra desenvolvida na questão anterior.

Assuma que apenas N processos podem ser executados em simultâneo.





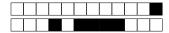
Questão 10 Considere um motor de pesquisa baseado em Inteligência Artificial que dada a interrogação de utilizadores (argumento prompt) responde em formato textual. Este motor é composto por um conjunto encadeado de programas.

- Para cada pedido, o ficheiro executável filter (*i.e.*, filter prompt>) seleciona ficheiros que contêm conteúdo relacionado com a interrogação, escrevendo em formato binário para o stdout os seus caminhos.
- Entretanto, o ficheiro executável execute (i.e., execute prompt>) lê do stdin os caminhos dos ficheiros filtrados, um a um, e gera conteúdo em binário de acordo com a interrogação do utilizador, o qual é escrito no stdout. Assuma que o conteúdo em binário gerado é sempre inferior a PIPE_BUF bytes.
- O ficheiro executável merge (i.e., merge prompt>) lê do stdin conteúdo binário resultante de interrogações e gera uma resposta para o utilizador em formato textual (escrita no stdout).
- 1. Escreva o programa SOGPT de forma a implementar o comportamento acima descrito. Otimize a fase de execute desencadeando o seu processamento concorrente com N processos.
- 2. Considere o programa cliente em baixo (search_prompt) que recorre ao pipe com nome fifo_server para enviar um pedido de pesquisa ao programa servidor. O pedido (estrutura Req q) contém o pid do programa cliente e o prompt a ser processado pelo servidor. Por fim, o programa cliente notifica o utilizador quando o pedido é completado, indicando-lhe o seu identificador.

```
int main (int argc, char * argv[]) {
2
        Req q;
3
        q.pid=getpid();
4
        q.prompt=strdup(argv[1]);
5
6
        char fifoc_name[30];
7
        sprintf(fifoc_name, "fifo_client_%d", q.pid);
8
        mkfifo(fifoc_name, 0666);
9
10
         int fds = open("fifo_server", O_WRONLY);
11
         write(fds, &q, sizeof(q));
         close(fds);
12
13
14
         int req_id;
         int fdc = open(fifoc_name, O_RDONLY);!
15
         read(fdc, &req_id, sizeof(int));
16
         printf("Request %d completed\n", req_id);
17
18
         close(fdc);
19
20
         unlink(fifoc_name);
21
         return 0;
22
     }
```

- (a) Escreva o programa servidor que, através do pipe com nome fifo_server, deverá receber pedidos de pesquisa de clientes (i.e., do programa search_prompt) e executar os mesmos sequencialmente, utilizando o programa SOGPT. Para cada pedido, o servidor deve atribuir-lhe um identificador único, a ser enviado ao cliente assim que o mesmo termina.
- (b) Com base na sua solução para a alínea (a), indique se a mesma funcionaria caso a abertura do pipe com nome fifoc_name (linha 15) fosse feita logo após a criação do mesmo (linha 8). Justifique a sua resposta.





Questão 11 Considere o seguinte código-fonte de um programa e selecione as afirmações verdadeiras:

```
1
    int i=0;
2
    while(i<10) {
3
         if (fork()==0) {
4
             op(i);
5
              i++;
              _exit(0);
6
7
         } else {
8
             i++;
         }
9
     }
10
```

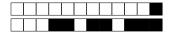
- A operação op(...) no máximo executa 5 vezes.
- B Só é executado op(i) quando i é par.
- C A operação op(...) no máximo executa 10 vezes.
- D A operação op(...) nunca é executada.
- E Nenhuma das respostas apresentadas está correta.

Questão 12 Considere o seguinte código-fonte de um programa e selecione as afirmações verdadeiras:

```
int main(int argc, char * argv[])
1
2
    {
3
        for (int i = 0; i < 3; i++) {
4
             if (fork() == 0) {
5
                 printf("processo %d\n", i);
6
                 func1();
                 _exit(0);
7
8
             } else {
                 wait(NULL);
9
10
                 printf("processo %d terminou\n", i);
11
              }
         }
12
13
14
         return 0;
     }
15
```

- A O programa cria processos que executam a func1() sequencialmente.
- B O programa cria processos que executam a func1() concorrentemente.
- C O output esperado é:
 processo 0
 processo 1
 processo 2
 processo 2 terminou
 processo 1 terminou
 processo 0 terminou

 D O output esperado é:
 processo 0
 - processo 0
 processo 0 terminou
 processo 1
 processo 1 terminou
 processo 2
 processo 2 terminou
- |E| Nenhuma das respostas apresentadas está correta.



Questão 13 Considere uma definição de struct Matricula e o seguinte código-fonte:

```
1
    int main(int argc, char * argv[])
2
         int fd = open("matriculas.bin", O CREAT | O RDWR, 0666);
3
4
         Matricula m;
           (\dots)
6
         if (read(fd, &m, sizeof(Matricula)) < 0) {</pre>
7
             perror("read");
8
             exit(1);
9
10
          write(1, m.matricula, strlen(m.matricula));
11
          close(fd);
12
          return 0;
     }
13
```

Considere um ficheiro binário matriculas.bin com 50 registos sequenciais de matrículas, numerados de 1 a 50. Selecione as opções que completam o código acima, i.e., que substituiem a linha de código 5 identificada por (...), de modo a que a variável m contenha o décimo segundo registo, após a leitura.

```
A lseek(fd, sizeof(Matricula) * 12, SEEK_SET);
B lseek(fd, sizeof(Matricula), SEEK_SET);
C lseek(fd, sizeof(Matricula) * 11, SEEK_SET);
D lseek(fd, -sizeof(Matricula) * 11, SEEK_END);
E Nenhuma das respostas apresentadas está correta.
```

Questão 14 Considere os seguintes pseudo-códigos para um programa cliente que envia mensagens para um programa servidor.

Cliente:

```
1
         int fd = open("fifo", O_WRONLY);
    2
         write(fd, ...);
    3
         close(fd);
Servidor:
         char buf[...];
    1
         int fd = open("fifo", O_RDONLY);
    2
         while (read(fd, buf, ...) > 0){
    3
    4
             write(1,buf, ...);
    5
         }
    6
         close(fd);
```

Considere que a comunicação entre cliente e servidor é efetuada através de um pipe com nome, o qual foi criado anteriormente com o nome fifo. Assuma também que o programa servidor é sempre executado antes do programa cliente. Indique quais das seguintes afirmações são verdadeiras.

- Após um cliente terminar a sua execução, e caso não exista mais nenhum cliente a executar, o servidor termina também o seu programa e não atende mais clientes.
- B Após um cliente terminar a sua execução, caso estejam outros programas clientes a executar e escrever mensagens para o servidor, o servidor termina também o seu programa, não atendendo estes clientes.
- C Após um cliente terminar a sua execução, e caso não exista mais nenhum cliente a executar, o servidor continua a executar o seu programa e a atender outros clientes que se possam ligar ao mesmo posteriormente.
- D Após um cliente terminar a sua execução, caso estejam outros programas clientes a enviar mensagens para o servidor, o servidor continua a executar o seu programa e a atender outros clientes até que todos tenham terminado.
- |E| Nenhuma das respostas apresentadas está correta.



Questão 15 Considere o seguinte código-fonte e selecione as afirmações verdadeiras:

```
1
        int pfd[2][2];
   2
        pipe(pfd[0]);
   3
   4
        if(fork()==0){
   5
            close(pfd[0][0]);
   6
            dup2(pfd[0][1], 1);
   7
            close(pfd[0][1]);
   8
            execlp("cat", "cat", "/etc/passwd", NULL);
            _exit(0);
   9
         }
   10
   11
   12
         close(pfd[0][1]);
   13
         pipe(pfd[1]);
   14
         if(fork()==0){
   15
   16
             close(pfd[1][0]);
   17
             dup2(pfd[0][0], 0);
   18
             close(pfd[0][0]);
   19
             dup2(pfd[1][1], 1);
   20
             close(pfd[1][1]);
   21
             execlp("sort", sort", NULL);
             _exit(0);
   22
   23
         }
   24
          (...)
   26
         if(fork()==0){
   27
   28
             dup2(pfd[1][0], 0);
   29
             close(pfd[1][0];
             execlp("wc", wc", -1, NULL);
   30
   31
              _exit(0);
   32
         }
   33
         close(pfd[1][0]);
A O seguinte código está em falta (linha 25)
             close(pfd[1][1]);
         1
             close(pfd[1][0]);
B O seguinte código está em falta (linha 25)
             close(pfd[0][0]);
             close(pfd[1][1]);
C O seguinte código está em falta (linha 25)
             close(pfd[0][0]);
         2
             close(pfd[1][0]);
```

- D O programa emula o comando cat etc/passwd | sort | wc -l
- E Nenhuma das respostas apresentadas está correta.