********** //**********

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUGESTÃO DE APRESENTAÇÃO DE RESPOSTA:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#1 Isto é uma questão?

@1 Isto é a resposta à questão #1.

%%%%%%%%

ALÍNEAS:

%%%%%%%%

#2 Isto é uma questão? a) isto é uma alínea desta questão?; b) isto é outra alínea desta questão?;

@2 Isto é a resposta à questão #2.

@2a Isto é a resposta à alínea a) da questão 2.

@2b Isto é a resposta à alínea b) da questão 2.

********** //**********

Semáforos

#1 Complete o código seguinte para garantir que é sempre impresso o valor correto (3000000). Utilize os semáforos definidos na norma POSIX.

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int  cont  = 0;




void  *worker(void *arg) {

int  n, c;

n = (int)arg; c = 0;



for(  i= 0; i < n; i++  )



c ++;



cont  = cont  + c;



}



int  main(int argc, char  *argv[]) {

pthread_t p1,  p2,  p3;
```

```
pthread_create(&p1, NULL, worker, (void*)1000000); pthread_create(&p2, NULL, worker,
(void*)1000000); pthread_create(&p3, NULL, worker, (void*)1000000);

pthread_join(p1, NULL); pthread_join(p2, NULL); pthread_join(p3, NULL);


printf("%d\n", cont);

return 0;

}
```

RESPOSTA: (provavelmente isto mas não tenho certeza)

```c
#include <semaphore.h>   //ESTA

int  cont  = 0;
sem_t a;


void  *worker(void *arg) {
int  n, c;
n = (int)arg; c = 0;

sem_wait(a); // esta
//-----------------------------------
for(  i= 0; i < n; i++  ) { //esta


c ++;
}               // esta
//-----------------------------------
cont  = cont  + c;
sem_post(a); // esta

}


int  main(int argc, char  *argv[]) {
pthread_t p1,  p2,  p3;

sem_init(&a, 0, 1); // ESTA

pthread_create(&p1, NULL, worker, (void*
pthread_join(p1, NULL); pthread_join(p2,

printf("%d\n", cont);
return 0;
}
```

#2 Complete o programa seguinte por forma a garantir que é sempre impresso o valor 24. Deve

utilizar apenas as operações sobre semáforos definidas na norma POSIX

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

```
int  y = 5;
int  x;

void  *worker1(void *arg) {




x = y - 1;




}
void  *worker2(void *arg) {




y = y * y;
}

int  main(int argc, char  *argv[])
{
pthread_t p1,  p2;




pthread_create(&p1, NULL, worker1, NULL); pthread_create(&p2, NULL, worker2, NULL);
pthread_join(p1, NULL); pthread_join(p2, NULL);

printf("%d\n", x);
return 0;
}
```

#3 Considere um mecanismo de sincronização entre processos caracterizada pelas operações seguintes:

- CriarBarreira(Nome, N) define um barreira de N processos
- AguardarBarreira(Nome) bloqueia o processo que invoca a operação até que N

processos tenham invocado esta operação.

Usando semáforos, escreva o código para AguardarBarreira. Suponha que a operação

CriarBarreira criou previamente todas as estruturas de dados necessárias.

#5 Apresente o pseudo-código das acções efectuadas pelas seguintes operações sobre semáforos, segundo a definição de Dijkstra, em que 'sem' indica um semáforo. Deve descrever todas as estruturas de dados auxiliares ou outras operações básicas de que necessite. Assuma que a indivisibilidade do código que apresentar está garantida:

a) Operação Wait(sem)

b) Operacão Signal(sem)

#6 Considere o problema dos leitores e escritores, processos concorrentes com acesso a uma base de dados, através das operações ler(R) e escrever(R) em que R designa um valor lido ou escrito. Admita que estas operações, que processam o acesso aos registos da base de dados, já estão implementadas, mas não controlam as interferências devidas aos acesso concorrentes dos múltiplos clientes leitores e escritores.

a) Baseando-se em semáforos, segundo a definição de Dijkstra, e em comunicação por memória

partilhada, apresente o pseudo-código das acções seguintes:

pedir_ler() pedir_escrever() terminar_ler() terminar_escrever()

que devem ser invocadas respectivamente, pelos leitores e escritores, antes e depois de poderem começar a ler ou a escrever. Note que as operações pedir_ler() e pedir_escrever() devem ser bloqueantes, caso o leitor ou o escritor não possa prosseguir por a base de dados estar correntemente ocupada por outros processos.

b) Explique quais as propriedades da solução que apresentou em a), do ponto de vista da justiça

(fairness) no tratamento dos pedidos dos leitores e escritores.

Gestão de memória

#7 Considere um sistema operativo (SO) que suporta múltiplos utilizadores e em que cada utilizador pode lançar mais do que um processo. Este SO gere a memória física com base em

partições criadas dinamicamente com dimensão igual à da imagem do processo que vai ser criado. Explique porque é que este sistema de gestão de memória está sujeito ao problema da fragmentação externa.

Resposta: Este sistema está sujeito a fragmentação externa pois a medida que os processos forem terminados vão deixando espaços de memória pequenos (buracos) impedindo a criação de novos processos pois não tem espaço para alocar memória devido a estes buracos.

Resposta(complementar ou mais correcta): À medida que os processos vao sendo criados, vao sendo deixados buracos na memoria, esses buracos podem não ser grandes o suficiente para alocar um novo programa, e por isso o novo programa deve ser fragmentado externamente para caber nesses espaços (buracos) de memoria

#8 Um CPU emite endereços virtuais com 24 bits. A unidade de gestão de memória (MMU) suporta paginação, tendo cada página 2048 (211) bytes. Diga de que forma é que feita a conversão dos endereços virtuais em endereços físicos. Indique, justificando, quantas entradas tem a tabela de páginas de um processo.

#9 Considere uma unidade de transformação de endereços baseada em páginas e em que o hardware mantém para cada página P o bit de referência R. R[P] está inicialmente a 0; sempre que é feita uma referência à página P, R[P] é colocado a 1. Há uma instrução máquina privilegiada que coloca R[P] a 0. Admita que sobre este hardware, o sistema operativo implementa uma estratégia de substituição de páginas de 2ª hipótese (2nd chance page replacement algorithm). Este algoritmo é usado sempre que é preciso escolher uma página V para ser substituída. Explique porque é que a página V escolhida é uma página que não é referenciada há muito tempo.

Resposta: Como o algoritmo indica este dá a todas as páginas uma 2oportunidade dando prioridade fazendo um balanço entre o espaço e o tempo de processamento. Sendo por isso escolhida a pagina V pois já tinha sido previamente referencia logo não e necessário fazer o processamento(longo) todo de novo estando pronta para ser swaped.

Resposta(complementar ou mais correcta): O algoritmo escolhe uma pagina que nao é referenciada à muito tempo porque é a que tem probabilidade menor de voltar a ser usada

#10 Explique como é que uma unidade de gestão de memória (MMU) com um registo base e um registo limite pode ser usada para efectuar recolocação de programas e para impedir que um processo tenha acesso a zonas da RAM que não fazem parte da sua imagem.

Resposta: O mmu avisa o cpu e o OS tenta encontrar um espaço livre de ram para meter o seu registo base se não houver espaço livre na ram tenta procurar uma page existente usando um algoritmo de replacement. O mmu garante a protecção da ram devido a sua memoria virtual

(registo base) transferindo memória fragmentada (pedaços livres e ocupados de memória todos juntos ou seja dados perigosos) para esta memória virtual evitando assim a sua extensão para a RAM.

Resposta(complementar): O MMU calcula a zona de a ram que se pode aceder atraves da base e do offset, não deixa aceder a um **espaço < base** ou a um **espaço >(base+offset).**

#11 Um CPU emite endereços virtuais com 24 bits. A unidade de gestão de memória (MMU) suporta paginação, tendo cada página 8192 (213) bytes. Diga de que forma é que feita a conversão dos endereços virtuais em endereços físicos. Indique, justificando, quantas entradas tem a tabela de páginas de um processo.

Resposta :A conversão e feita a traves do registo de recolocação (registo base), os conteúdos deste são adicionados a cada endereço virtual para determinar o endereço físico correspondente . Entradas??

213 bytes = 213*8bits = 1704 como 1704 é maior que 2^10bits, é preciso 2^11 bits portanto: 2^(24-11) = 2^13 = 8192 entradas

#12 Considere um sistema de gestão de memória em que a MMU (Memory Management Unit) suporta uma tabela de páginas, em que cada página tem um bit de validade V. Sobre este hardware, o sistema operativo suporta paginação a pedido. Descreva as acções efectuadas pelo SO quando recebe uma interrupção da MMU assinalando que o processo A tentou referenciar a sua página virtual P, tendo a entrada P da tabela de páginas do processa A o bit de validade a 0.

Resposta: O OS simplesmente aborta? Se a pagina não é valida, também não tem nenhum endereço fisico válido para aceder (deve estar incompleta)

#13 O que entende por um sistema de gestão de memória central que funciona com paginação a pedido? Explique como é que este sistema permite executar programas cuja dimensão é superior ao tamanho da memória física.

Resposta: Este sistema de gestão de memória central permite ao CPU guardar e retirar data de um armazenamento secundário para usar na memória principal (usando páginas).Permite pois o tamanho da memória continua a ser inferior ao potencial espaço de endereçamento da memória física , por definição o adress map reserva grandes regiões para I/O operações dái grande parte da memória (virtual) não pode ser usado.

Resposta: Para executar programas que superam a memoria fisica, é necessário "roubar"/utilizar uma page de já em uso , gravam se os seus conteudos e carregam-se os da nova pagina que queremos criar. Se essa que substituimos for chamada, é preciso voltar a por o seu conteudo original, e encontrar outra para "roubar"

#14 A gestão da memória física por partições apresenta vários inconvenientes. Um deles é a fragmentação da memória.

a) A fragmentação pode ser interna ou externa. Explique o que significa cada um destes termos

?

Resposta: A fragmentação interna deriva de alocação de memória excessiva, ficando com um excesso de memória alocada que não vai ser usada nem pode ser utilizada , enquanto que a fragmentação externa é definida pela grande dispersão de pequenos blocos de memória entre blocos ocupados de memória alocada tornando impossível usar estes blocos pois são demasiados pequenos para serem alocados.

b) Num sistema com partições em número e dimensões fixas, que tipo de fragmentação ocorre?

Justifique.

Resposta: Fragmentação interna se um programador escolhe um bloco pequeno demais o sistema não o deixa alocar essa memoria se for muito grande para o necessário vai alocar memória a mais do que precisa daí a fragmentação interna.

c) E num sistema com partições variáveis?

Resposta: Fragmentação externa, ao serem modificadas (apagadas removidas etc..) estas partições deixam buracos no meio da memória alocada. (imaginem um ficheiro de texto tem o cabeçalho parte do meio e fim ao modificarem imaginemos a parte do meio ficam com os tais buracos de memória)->fragmentação externa.

#15 No LINUX a gestão de memória utiliza uma MMU baseada em páginas. Quando é feito um fork() como é preenchida a tabela de páginas do novo processo?

Resposta: Cria um processo com uma cópia da tabela de paginas que chamou o fork, esta tp são todas cópias do mapeamento partilhado das páginas físicas entre o pai e o filho

#16 Explique os inconvenientes dos sistemas de gestão de memória física baseados em partições contíguas. Explique de que forma é que os sistema de gestão de memória física baseados em páginas contribuem para resolver esses problemas.

Resposta: Em partições continuas (ms-dos) pode sofrer de fragmentação externa (exemplo do Windows ficam sempre bocados de memória que não podemos usar quando modificamos ou apagamos ficheiros) e é sempre necessário declarar o tamanho na altura da criação do disco. Enquanto que os sistemas em páginas elimina a fragmentação externa pois não cria segmentos gigantes e ao mesmo tempo não adressa memória que não vai ser usada (normalmente não existem páginas do tamanho de um segmento em partições continuas)

#17 Considere um sistema de gestão de memória baseado em páginas, suportado em paginação a pedido.

a) É possível executar um programa cujo imagem tenha uma dimensão superior à da memória física? Se sim, em que condições?

   Resposta: Sim, pois na maioria dos casos a memoria principal e muito mais pequena que a memória de endereçamento virtual.

b) Neste ambiente, explique, em detalhe, o que ocorre quando o processo corrente referencia uma página que não está em memória.

   Resposta: Quando referencia uma página que não está em memória , procura ou espaço livre de ram para usar , caso contrário se não houver ram disponível utiliza um 2nd replace algorthum para tratar dessa página (já descrito em cima).

#18 Suponha um S.O. que executa numa máquina que tem uma MMU baseada em páginas. A MMU contém um "Translation Lookaside Buffer" (TLB). Explique qual a utilidade do TLB. Diga o que acontece ao TLB sempre que se comuta de processo.

Resposta: O tlb é uma cache que a mmu usa para acelerar o endereçamento virtual .Quando se comuta de processo vai procurar se encontra o adress pedido no tlb , se não o encontrar vai ler os conteúdos de memoria de várias localizações ate encontrar este adress e envia o para a tlb.

#19 Num dado S.O. a gestão de memória utiliza uma MMU baseada em páginas e é utilizada paginação a pedido. Explique em que consiste a situação de "trashing". Diga também de que forma de pode detectar que um dado sistema entrou em "trashing".

Resposta: Trashing acontece quando a memoria virtual entra numa constante troca de informações entre a memoria e o disco(a procura de uma page) fazendo com que o disco entre em colapso ate que a o adress(page) seja encontrado.O sistema entrou em trashing quando o disco demora muito tempo a encontrar o endereço em causa. Podendo até mesmo entrar em colapso e perder performance.

Sistema de ficheiros e dispositivos de entrada / saída

#20 No sistema operativo Linux, um processo lançado pelo utilizador U executa a chamada ao sistema

int f = open("/home/bruno/x.c", O_RDONLY);

Explique como é que o sistema operativo verifica que o ficheiro indicado existe.

Resposta: O SO considera que o ficheiro existe até que todos os processos que tenham um file descriptor deste sejam apagados. Ou seja só quando nenhum processo tenha um fd deste ficheiro o SO diz que o ficheiro não existe.

#21 Considere um disco que depois de formatado tem 65536 (216) blocos e em que cada bloco tem 4096 (212) bytes; os endereços dos blocos têm 16 bits (2 bytes). Suponha que neste disco foi colocado um sistema de ficheiros que usa uma estratégia de atribuição de espaço em disco indexada, em que os blocos atribuídos a um ficheiro são definidos por 8 endereços de blocos. Os endereços de 0 a 6 são endereços directos e o endereço 7 é o endereço de um bloco que contém endereços de blocos. Diga, justificando, qual é o tamanho máximo, usando esta estratégia de atribuição, que um ficheiro pode ter?

#22 Considere a seguinte sequência de chamadas ao sistema

(1) int f = open("/x.log", O_WRONLY | O_CREAT);

(2) write(f, '123456', 6);

(3) close(f);

Para cada uma das chamadas ao sistema explique que consultas e alterações são feitas na área de meta-dados do disco que suporta o sistema de ficheiros em que está o ficheiro x.log. Suponha que o ficheiro não existia antes da chamada ao sistema (1) e que o processo que faz as chamadas tem permissões para escrever na directoria raiz do sistema.

#23 Considere um disco que depois de formatado tem 224 blocos e em que cada bloco tem 4096 (212) bytes; os endereços dos blocos têm 32 bits (4 bytes). Suponha que neste disco foi colocado um sistema de ficheiros que usa uma estratégia de atribuição de espaço em disco indexada, em que os blocos atribuídos a um ficheiro são definidos por 8 endereços de blocos. Os endereços 0 a 5 são endereços directos e os endereço 6 e 7 são os endereços de dois blocos que contêm endereços de blocos. Diga, justificando, qual é o tamanho máximo, usando esta estratégia de atribuição, que um ficheiro pode ter?

#24 Considere o sistema de ficheiros do MS-DOS que é baseado na FAT (File Allocation Table). Qual dos métodos de atribuição de espaço em disco aos ficheiros estudados é usado? Justifique. Indique, justificando, qual é a máxima dimensão de um disco lógico suportado por este método.

#25 Considere o sistema de ficheiros EXTENDED 2 do Linux. Descreva os vários passos efectuados quando se apaga um ficheiro, dado o seu nome.

#26 Indique os passos realizados quando se efectua uma chamada ao sistema que realiza uma operação de saída de dados síncrona sobre um periférico tipo bloco. Admita que o periférico assinala a conclusão da operação através de uma interrupção.

#27 Considere o sistema de ficheiros do UNIX. Explique o que se passa, em termos das estruturas de dados que descrevem o estado do sistema de ficheiros quando se dá o comando

mv /d1/f1 /d1/f2 .

Suponha que antes de dar o comando a directoria d1 e o ficheiro f1 existem e o ficheiro f2 não existe

#28 Considere um S.O. da família UNIX. Explique para que servem as chamadas ao sistema open() e close(). Concretize para o caso de um ficheiro normal e de um ficheiro especial (periférico). Note que não se pretende uma descrição de como é que estas chamadas são implementadas.

#29 Considere o sistema de ficheiros EXT2 do LINUX. Descreva os vários passos efectuados quando, no interpretador de comandos, se escreve mv nome_lógico_1 nome_lógico_2.

#30 Compare as formas de atribuição de espaço em disco a ficheiros utilizadas pelo sistema de ficheiros FAT (Windows) e pelo UNIX. Descreva as principais estruturas utilizadas por cada um deles. Indique as vantagens e os inconvenientes dos dois métodos.