LISTA DE EXERCÍCIOS

SISTEMAS OPERACIONAIS I

Questões referentes a Introdução sobre S.O

1. Quais são as funções principais de um S.O.?

2. O que é multiprogramação? Dê duas razões que justifiquem a utilização deste conceito.

3. Explique como um Sistema de Computação se divide.

4. Explique os conceitos básicos sobre arquivos, diretórios, pipe e Shell.

5. Para um programador, uma chamada de sistema se parece com qualquer outra chamada a um procedimento de biblioteca. É importante que um programador saiba quais procedimentos de biblioteca resultam em chamadas ao sistema? Sob quais circunstâncias e por quê?

6. No Slide 27 da Introdução mostra várias chamadas de sistema ao Sistema Unix não tem equivalência na API do Win32. Para cada chamada relacionada que não tenha equivalência Win32, quais são as conseqüências para o programador em converter um programa Unix para executar no Windows?

7. Por que é necessária uma tabela de processos em sistema de tempo compartilhado? Essa tabela é essencial também em sistemas de computador pessoal (PC), nos quais existe apenas um processo, que detém o comando de toda a maquina até que ele termine?

8. O Multiprocessamento permite a Re-configuração e o Balanceamento do sistema. Fale sobre estas características.

9. Por que aplicações que necessitam de execução em tempo real não são indicadas para rodar em um sistema de tempo compartilhado?

10. Cite os principais acontecimentos que ocorreram durante a 1a, 2a, 3a e 4a na evolução dos sistemas operacionais.

11. Explique os tipos de SO: Monoprogramado, Multiprogramado, Multiprocessado e em tempo real.

12. sobre Arquiteturas de Sistema Operacionais explique cada um individualmente e depois faça um comparativo entre elas em uma tabela.

Introdução a Processos

1. Explique:

    A. Todos os passos da mudança de contexto entre dois processos;

B. Quais os estados que um processo passa durante se ciclo de vida e como o S.O. gerencia essa mudança de estado.

2. Cite e explique como é formado/constituído um ambiente de um processo.

3. Explique como funcionam os escalonadores de longo termo e de curto termo.

4. Explique o que é Bloco de Controle do Processo (PCB) e qual a sua utilização em sistema multiprogramáveis de tempo compartilhado.

5. Pensando em estrutura de dados, explique como as filas de pronto e bloqueado são constituídas.

6. Faça uma pesquisa no simulador SOSIM e veja quais são os tipos de processos disponíveis. Explique cada um deles.

7. Sobre a troca de informações entre processos, explique as formas de comunicação interprocessos - IPC. Dê Exemplos:

Criação de Processos com uso de Fork

1) Sobre criação de processos explique:
a) Fork, Exec e Wait;
b) Quais são os valores retornado pelo fork e para que servem.
c) Como funciona a criação de processos filhos, levando em consideração o que é herdado ou não do processo pai.
d) Qual a função da Chamada Copy-on-Write no Linux.
e) Explique o que os termos órfãos e zombie em Linux. Como o SO lida com eles e quais as consequências deles. Como podemos diferenciar um do outro no SO.

2) Para os códigos abaixo apresente o gráfico de precedência de execução.

a)

```
int main()
{
        fork()
        printf("hello\n");
}
```

b)

```
int main()
{
        fork()
        fork()
        printf("hello\n");
}
```

c)

```
int main()
{
        fork()
        fork()
        fork()
        printf("hello\n");
}
```

d)

```
int main()
{
        for (i=0;i<2;i++)
                fork();
        printf("Ola\n")
}
```

e)

```
doit()
{
        fork()
        fork()
        printf("hello\n")
}
int main()
{
        doit()
        printf("Ola\n")
}
```

3) Qual é o output do seguinte programa? Monte o Gráfico de Precedência de execução destacando em cada processo o que está sendo impresso.

a)

```
int main()
{
        int pid, x = 4;
        pid = fork();

        if ( 0 == pid )
        {
                fork();
                x=x+2;
        }
        else
        {
                 x--;
        }
```

```
                printf(“x=%d\n”,x);
        }
```

b)

```
#include <stdio.h>
#include <stdlib.h>

int int main()
{
   char *args[] = {"/bin/ls", "-l", (char *) 0 };

    if (0 == fork() )
        printf("Filho\n");
    else
   {
        printf ("Lista diretorio:\n");
        execv ( "/bin/ls", args);
        printf(" Pai\n");
   }
   Printf ("Fim\n");
}
```

4) O que será escrito no arquivo e porque?

```
int main()
{
        FILE *fp=fopen(“out.txt”,”w”);

        fprintf (fp,“Bom Dia\n“); // aqui faltava o \n – faça um teste sem e com ele.

        if (0 == fork() )
        {
                fprintf( fp , “ Filho\n”);
                flcose(fp);
        }
        else
        {
                fprintf( fp,“ Pai\n”);
        }

}
```

5) O que será escrito na tela, em qual ordem e porque ?

```
int main()
{
        printf (“Bom Dia \n“); // aqui faltava o \n - – faça um teste sem e com ele.
```

```
        if (0 == fork() )
                printf(“ Filho\n”)
        else
                printf(“ Pai\n”);
}
```

6) Explique a Saída deste código:

```
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int main()
{
pid_t pid, pid2;
int estado;

pid = fork(); /* Cria um PROCESSO */
if (pid < 0)
{
        printf("Erro ao criar o processo\n");
        exit(-1);
}
else
if (pid >0) /* Código que só vai ser executado pelo Processo PAI */
{
        printf("Eu sou o PAI\n");
        pid2 = wait(&estado);
        printf("O filho com o pid %d terminou\n",pid2);
}
else /* Código que só vai ser executado pelo Processo FILHO */
printf("Processo Filho\n");
printf("REPETIDO\n");
}
```

Observação:
n = wait (estado) - obriga o pai a esperar pela terminação de qualquer processo. A n é associado o pid do primeiro processo filho que terminar e o respectivo valor de retorno é associado a estado

Se estado == -1 sabe-se que o filho terminou de forma inesperada, caso contrário, foi de forma correta.

O estado pode ser utilizada em outras funções visando buscar informações sobre o processo filho. No material extra existe informações sobre essas funções.

7- Considere a seguinte parte de um programa:

```
...
p = fork();
```

```
a = fork();

printf("Sistemas Operativos II\n");
...
```

a) Quantas vezes é apresentada "Sistemas Operativos II" ? Justifique.
b) Quantos processos são criados pelo programa?

8- Faça um programa que crie 2 processos e:

Escreve "Eu sou o pai" no processo pai.
Escreve "Eu sou o 1º filho" no primeiro filho.
Escreve "Eu sou o 2º filho" no segundo filho.

Monte o gráfico de precedência de execução destacando em cada processo o que está sendo impresso.

9 - Considere o seguinte pedaço de código em C:

```
…
for (i=0; i<4; i++ )
  pid = fork();

printf("SOP2\n");
…
```

a) Quantos processos são criados por este programa? Justifique.
b) Quantas vezes é apresentado "SOP2"? Justifique.
c) Monte o gráfico de precedência de execução destacando em cada processo o que está sendo impresso.

10 - Considere a seguinte parte de um programa:

```
main()
{
        ...
        pid2=0;
        for(i=0;i<2;i++)
        {

                pid=fork();
                if (pid==0)
                        pid2=fork();
                if (pid2==0)
                        printf("Exame de SO1\n");
        }
        ...
}
```

Quantas vezes é apresentada a frase “Exame de SO1” ? Justifique.
Monte o gráfico de precedência de execução destacando em cada processo o que está sendo impresso.

11 – Considere a seguinte parte de um programa:

```
main ()
{
    ...
    for(i=0; i<2: i++)
    {
        pid1=fork();
        pid2=fork();
        if (pid1 > 0)
            printf("Sistemas\n");
        if (pid2 == 0)
            printf("Operativos\n");
    }
    ...
}
```

Quantas vezes são apresentadas as palavras "Sistemas" e "Operativos"? Justifique.
Monte o Gráfico de Precedência de execução destacando em cada processo o que está sendo impresso.

Nota: Considere que a função fork() nunca devolve valores menores que zero.

12 - Tendo um array de inteiros com mil posições, crie um processo e faça o seguinte:

Os dois processos (pai e filho) fazem o seguinte cálculo: resultado[i] = dados[i]*4 + 20.
Cada processo é responsável pelo cálculo de 500 posições do array dados, colocando o resultado no array resultado.
Os resultados devem ser apresentados segundo a ordem do array (de 0 a 499 e depois de 500 a 999).

13 - Faça um programa que crie 6 processos e:

Cada processo escreve 200 números:

1º processo: 1 .. 200
2º processo: 201 .. 400
3º processo: 401 .. 600
4º processo: 601 .. 800
5º processo: 801 .. 1000
6º processo: 1001 .. 1200

O processo pai tem de esperar que todos os processos filhos terminem.

14 - Tendo um array de 1000 posições, faça um programa que crie 5 processos e:

Dado um número, procurar esse número no array.
Cada processo filho, procura 200 posições.
O processo que encontrar o número, deve imprimir a posição do array onde se encontra. Também deve "retornar" como valor de saída o número do processo (1, 2, 3, 4, 5).
Os processos que não encontrarem o número devem "retornar" como valor de saída o valor 0.
O processo pai tem de esperar que todos os filhos terminem e imprimir o número do processo onde esse número foi encontrado (1, 2, 3, 4, 5).

Nota: O array não tem números repetidos.

15 - Crie a seguinte função: char cria_gémeos(pid_t lista[2]) – esta função deve criar dois processos e devolver para o primeiro processo criado o caracter 'a' e para o segundo processo o caracter 'b'. Para o pai, a função devolve o caracter 'p' assim como os identificadores dos processos criados.

16 – Execute e explique o que está acontecendo

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>

int main ( void )
{
      pid_t pid,pid2;
      int estado;
      pid=fork();
      if(pid<0){
            printf("Erro\n");
            exit(-1);
      } else {
                  if(pid==0){
                        printf("Eu sou o 1º filho\n");
                          printf("Pid do 1º filho: %d\n", getpid());
                          sleep(5);
                          exit(1);
                  }
                     else
                     {
                          pid2=fork();
                          if(pid2<0){
                              printf("Erro\n");
                              exit(-1);
                          }
```

```
                        else
                        {
                            if (pid2==0) {
                                printf("Eu sou o 2º filho\n");
                                printf("Pid do 2º filho: %d\n", getpid());
                                exit(2);
                            } else {
                                 printf("Eu sou o Pai\n");
                                 printf("Pid do Pai: %d\n", getpid());
                                 pid = waitpid(pid, &estado,0);
                                 if(WIFEXITED(estado)){
                                      printf("Valor de saída do 1º filho %d\n",
WEXITSTATUS(estado));
                                 }
                                 pid2=waitpid(pid2,&estado,0);
                                 if(WIFEXITED(estado)){
                                     printf("Valor de saída do 2º filho %d\n", WEXITSTATUS(estado));
                                 }
                            }
                        }
                    }
                }
        }
  }
```

17) Indique quantas letras "X" serão impressas na tela pelo programa abaixo quando for executado com a seguinte linha de comando: a.out 4 3 2 1
Monte o Gráfico de Precedência de execução destacando em cada processo o que está sendo impresso.

```
int main(int argc, char *argv[])
{
        pid_t pid[10];
        int i;

        int N = atoi(argv[argc-2]);
        for (i=0; i<N; i++)
                pid[i] = fork();

        if (pid[0] != 0 && pid[N-1] != 0)
                pid[N] = fork();

        printf("X");

        return 0;
}
```

Escalonamento de Processos

1) O que é política de escalonamento de um sistema operacional?

2) Quais os principais critérios utilizados em uma política de escalonamento?

3) Diferencie os tempos de processador, espera, turnaround e resposta.

4) Diferencie os escalonamentos preemptivos e não-preemptivos.

5) Qual a diferença entre os escalonamentos FIFO e circular?

6) Descreva o escalonamento SJF e o escalonamento por prioridades.

7) Qual a diferença entre preempção por tempo e preempção por prioridade?

8) Resolva os seguintes escalonamentos utilizando FIFO. Calcule o tempo médio de escalonamento (espera e turnaround). Desenhe um diagrama ilustrando o escalonamento dos processos.

a) P1 = 30, P2 = 10, P3 = 5

b) P1 = 25, P2 = 5, P3 = 30

c) P1 = 5, P2 = 20, P3 = 30

d) Compare os três resultados e discuta a melhor opção.

9) Resolva os seguintes escalonamentos utilizando SJF. Calcule o tempo médio de escalonamento (espera e turnaround). Desenhe um diagrama ilustrando o escalonamento dos processos.

a) P1 = 6, P2 = 10, P3 = 8, P4 = 2 (Ordem de chegada p4, p1, p3, p2)

b) P1 = 8, P2 = 1, P3 = 18, P4 = 12 (Ordem de chegada p2, p1, p4, p3)

c) Compare os dois resultados e discuta a melhor opção.

10) Resolva os seguintes escalonamentos utilizando SJF preemptivo. Calcule o tempo médio de escalonamento (espera e turnaround). Desenhe um diagrama ilustrando o escalonamento dos processos.

a) P1 = 8, p2 = 9, p3 = 4, p4 = 2 (Ordem de chegada p1=0, p2=1, p3=2, p4=3)

b) P1 = 6, p2 = 9, p3 = 2, p4 = 4 (Ordem de chegada p1=0, p2=1, p3=2, p4=3)

c) Compare os dois resultados e discuta a melhor opção.

11) Resolva os seguintes escalonamentos utilizando Escalonamento Circular. Calcule o tempo médio de escalonamento (espera e turnaround). Desenhe um diagrama ilustrando o escalonamento dos processos.

a) P1 = 8, p2 = 9, p3 = 4, p4 = 2 (Quantum = 4)

b) P1 = 6, p2 = 9, p3 = 2, p4 = 4 (Quantum = 2)

c) Compare os dois resultados e discuta a melhor opção.

12) Resolva os seguintes escalonamentos utilizando Tempo Real..

a) Para um processo P1 com Períodos de Eventos = 50, 110, 150, 250, e tempo CPU = 35, 20, 10 e X. Qual é o valor de X para que o sistema seja escalonável.

b) Para um processo P2 com Períodos de Eventos = 110, 250, 350, e tempo CPU = 50, 30, 100. Acrescentar um 4o evento cujo período seja de 1 segundo. Qual é o valor de X para que o sistema seja escalonável.

Concorrência entre Processos

1. O que é condição de corrida?

2. Explique a diferença entre espera ocupada e bloqueio.

3. A solução com espera ocupada, usando a variável turn (estrita alternância nos slides), funciona quando estiver sendo usada em um sistema com dois processadores acessando memória comum?

4. Quais as principais diferenças entre semáforos e monitores.

5. Porque que é necessária a utilização de um conceito conhecido como inversão de prioridades?

6. A solução de Peterson apesar de resolver o problema da exclusão mutua possui um problema agravante da solução, qual é esse problema?

7. Quais são as 4 condições que devem ser garantidas para se evitar que dois ou mais processos estejam na sua região critica ao mesmo tempo, evitando condições de corrida? Explique cada uma delas.

8. No código exemplo do Sleep/Wakeup identificamos uma condição de corrida que foi a perda de um sinal WAKEUP. Na realidade existe uma segunda condição de corrida que pode acontecer. Qual é essa condição? (observe a variável count).

9. Não é difícil que um processador tenha uma instrução TSL e uma outra primitiva de sincronização, tal com semáforos ou monitores. A instrução TSL e as primitivas têm objetivos completamente diferentes e não competem entre si. Explique por quê?

10. Por que é necessária a utilização da exclusão mútua na concorrência entre processos?

11. Cite e explique dois problemas de sincronização que poderão ocorrer na comunicação entre processos.

12. Explique o que é Exclusão mútua e Região crítica.

13. O que é starvation e como podemos solucionar esse problema?

14. Qual o problema com a solução que desabilita as interrupções para implementar a exclusão mútua?

15. Em uma aplicação concorrente que controla saldo bancário em contas correntes, dois processos compartilham uma região de memória onde estão armazenados os saldos dos clientes A e B. Os processos executam concorrentemente os seguintes passos:

Processo 1 (Cliente A)

```
/* saque em A */
1a. x := saldo_do_cliente_A;
1b. x := x - 200;
1c. saldo_do_cliente_A := x;

/* deposito em B */
1d. x := saldo_do_cliente_B;
1e. x := x + 100;
1f. saldo_do_cliente_B := x;
```

Processo 2 (Cliente B)

```
/*saque em A */
2a. y := saldo_do_cliente_A;
2b. y := y - 100;
2c. saldo_do_cliente_A := y;

/* deposito em B */
2d. y := saldo_do_cliente_B;
2e. y := y + 200;
2f. saldo_do_cliente_B := y;
```

Supondo que os valores dos saldos de A e B sejam, respectivamente, 500 e 900, antes de os processos executarem, pede-se:

a) Quais os valores corretos esperados para os saldos dos clientes A e B após o término da execução dos processos?

b) Quais os valores finais dos saldos dos clientes se a sequência temporal de execução das operações for: 1a, 2a, 1b, 2b, 1c, 2c, 1d, 2d, 1e, 2e, 1f, 2f?

c) Utilizando semáforos, proponha uma solução que garanta a integridade dos saldos e permita o maior compartilhamento possível dos recursos entre os processos, não esquecendo a especificação da inicialização dos semáforos.

16. O problema dos leitores/escritores, apresentado a seguir, consiste em sincronizar processos que consultam/atualizam dados em uma base comum. Pode haver mais de um leitor lendo ao mesmo tempo; no entanto, enquanto um escritor está atualizando a base, nenhum outro processo pode ter acesso a ela (nem mesmo leitores).

```
VAR Acesso: Semaforo := 1;
Exclusao: Semaforo := 1;
Nleitores: integer := 0;

PROCEDURE Escritor;
BEGIN
        ProduzDado;
        DOWN (Acesso);
        Escreve;
        UP (Acesso);
END;
```

```
PROCEDURE Leitor;
BEGIN
        DOWN (Exclusao);
        Nleitores := Nleitores + 1;
        IF ( Nleitores = 1 ) THEN DOWN (Acesso);
        UP (Exclusao);
        Leitura;
        DOWN (Exclusao);
        Nleitores := Nleitores - 1;
        IF ( Nleitores = 0 ) THEN UP (Acesso);
        UP (Exclusao);
        ProcessaDado;
END;
```

a) Suponha que exista apenas um leitor fazendo acesso à base. Enquanto este processo realiza a leitura, quais os valores das três variáveis?

b) Chega um escritor enquanto o leitor ainda está lendo. Quais os valores das três variáveis após o bloqueio do escritor? Sobre qual(is) semáforo(s) se dá o bloqueio?

c) Chega mais um leitor enquanto o primeiro ainda não acabou de ler e o escritor está bloqueado. Descreva os valores das três variáveis quando o segundo leitor inicia a leitura.

d) Os dois leitores terminam simultaneamente a leitura. É possível haver problemas quanto à integridade do valor da variável nleitores? Justifique.

e) Descreva o que acontece com o escritor quando os dois leitores terminam suas leituras. Descreva os valores das três variáveis quando o escritor inicia a escrita.

f) Enquanto o escritor está atualizando a base, chagam mais um escritor e mais um leitor. Sobre qual(is) semáforo(s) eles ficam bloqueados? Descreva os valores das três variáveis após o bloqueio dos recém-chegados.

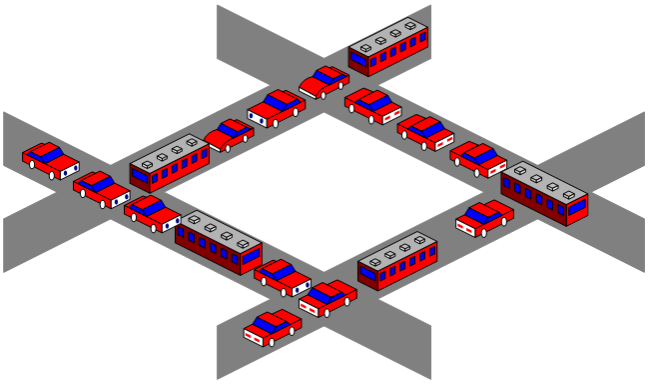
g) Quando o escritor houver terminado a atualização, é possível prever qual dos processos bloqueados (leitor ou escritor) terá acesso primeiro à base?

h) Descreva uma situação onde os escritores sofram starvation (adiamento indefinido).

Deadlock

1. O que é deadlock, quais as condições para obtê-lo e quais as soluções possíveis?
2. Liste três exemplos de deadlocks que não estejam relacionados com um ambiente de sistema de computação.
3. É possível ter um deadlock envolvendo apenas um único processo? Explique sua resposta.

4. Considere o deadlock de trafego da figura abaixo. Mostre que as quatro condições necessárias para o deadlock de fato estão presentes neste exemplo.



5. Considere um sistema que consista de 4 recursos do mesmo tipo que são compartilhados por 3 processos, cada qual precisando no máximo de 2 recursos. Mostre que o sistema está livre de deadlocks.
6. Explique uma situação onde um grafo com um ciclo não possui deadlock. Lembre que ciclos geram deadlocks.
7. Quais são os métodos para tratar deadlocks.
8. Explique a diferença entre prevenção e impedimento de deadlocks.
9. Quais são os métodos para prevenir deadlocks.
10. Explique a diferença entre um grafo de alocação de recursos de um grafo de espera.
11. Explique as diferenças entre os métodos de recuperação de deadlocks.

Threads

1. Enumere as principais vantagens e desvantagens das threads em relação aos processos.
2. Explique as principais diferenças entre threads no modo usuário e threads no modo kernel;
3. Quais elementos são considerados privados de um thread durante seu ciclo de vida e quais são compartilhados?
4. Explique como funciona a mudança de contexto entre threads gerenciados pelo Kernel e gerenciados pelo usuário.

5. Utilize o exemplo de criação pelo método A – Threads e o implemente utilizando por interface (B).

```
public class MyThread extends Thread {
  private String whoami;
  private int delay;

public MyThread(String name, int d) {
  whoami = name;
  delay = d;
}
```

```
 public void run() {
   try {
      sleep(delay);
   } catch(InterruptedExcetion e) {}
   System.out.println("Hello, this is
 "+whoami+" ! ");
   }
 }
```

```
public class TestThreads {
  public static void main(String[] args) {
     MyThread t1, t2, t3;
     t1 = new MyThread("First", 1000);
     t2 = new MyThread("Second", 500);
     t3 = new MyThread("Third", 2000);

     t1.start();
          t2.start();
          t3.start();
       }
     }
```

execução provável:

- Hello, this is Second!
  Hello, this is First!
  Hello, this is Third!

6. O que é uma Java thread?
7. Para que serve? Aplicações?
8. Quando deve ser usada as threads?
9. Desvantagens das threads?
10. Quantos fluxos de execução possui uma thread?
11. Outros atributos de uma Java thread?
12. Quem escalona as Java threads?
13. Qual o tipo de escalonamento?
14. Quais os passos de criação por herança?
15. Como/quando uma Java thread é ativada?
16. Quais os passos de criação por interface?
17. Qual a vantagem de criação por interface?
18. Uma Java thread pode ter métodos do usuário?

GABARITO

Escalonamento de Processos

8) FIFO

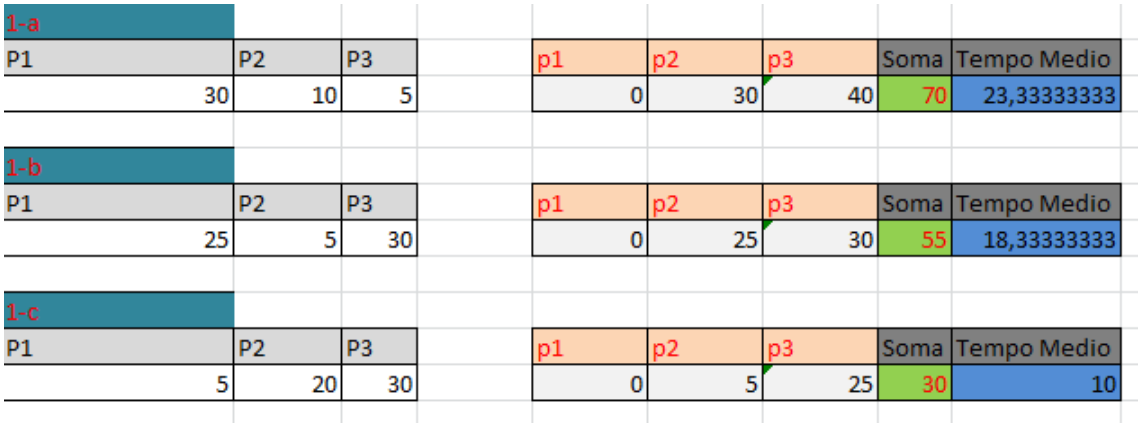
a) P1 = 30, P2 = 10, P3 = 5

b) P1 = 25, P2 = 5, P3 = 30

c) P1 = 5, P2 = 20, P3 = 30

d) Compare os três resultados e discuta a melhor opção.

Tempo de Espera



1-a							
P1	P2	P3	p1	p2	p3	Soma	Tempo Medio
30	10	5	0	30	40	70	23,33333333

1-b							
P1	P2	P3	p1	p2	p3	Soma	Tempo Medio
25	5	30	0	25	30	55	18,33333333

1-c							
P1	P2	P3	p1	p2	p3	Soma	Tempo Medio
5	20	30	0	5	25	30	10

Tempo de Tournaround

9) SJF

a) P1 = 6, P2 = 10, P3 = 8, P4 = 2 (Ordem de chegada p4, p1, p3, p2)

b) P1 = 8, P2 = 1, P3 = 18, P4 = 12 (Ordem de chegada p2, p1, p4, p3)

c) Compare os dois resultados e discuta a melhor opção.

2-a

	P1	P2	P3	P4		p1	p2	p3	p4	Soma	Tempo Medio
Time de CPU	6	10	8	2		2	16	8	0	26	6,5
Ordem Chegada	p4	p1	p3	p2		26					

2-b

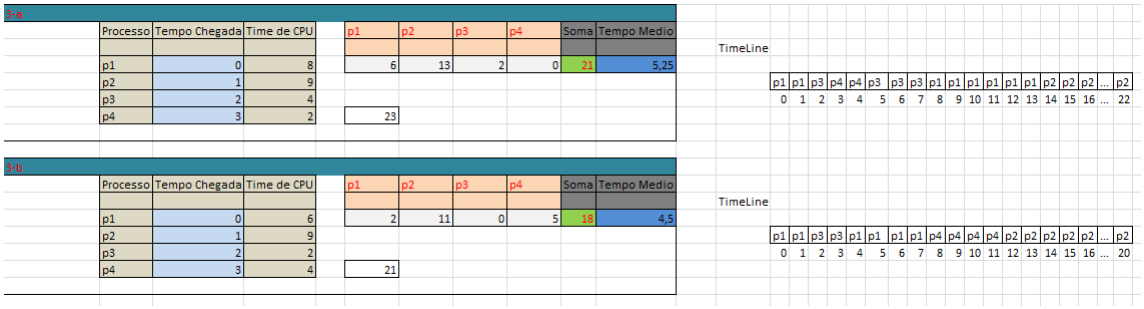
	P1	P2	P3	P4		p1	p2	p3	p4	Soma	Tempo Medio
Time de CPU	8	1	18	12		1	0	21	9	31	7,75
Ordem Chegada	p2	p1	p4	p3		39					

10) SJF preemptivo

a) P1 = 8, p2 = 9, p3 = 4, p4 = 2 (Ordem de chegada p1=0, p2=1, p3=2, p4=3)

b) P1 = 6, p2 = 9, p3 = 2, p4 = 4 (Ordem de chegada p1=0, p2=1, p3=2, p4=3)

c) Compare os dois resultados e discuta a melhor opção.

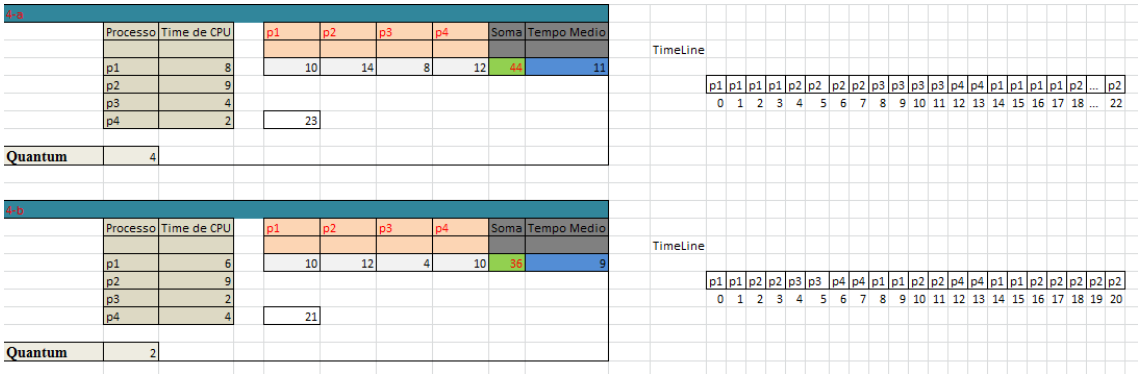


11) Escalonamento Circular

a) P1 = 8, p2 = 9, p3 = 4, p4 = 2 (Quantum = 4)

b) P1 = 6, p2 = 9, p3 = 2, p4 = 4 (Quantum = 2)

c) Compare os dois resultados e discuta a melhor opção.



12) Tempo Real..

a) Para um processo P1 com Períodos de Eventos = 50, 110, 150, 250, e tempo CPU = 35, 20, 10 e X. Qual é o valor de X para que o sistema seja escalonável.

(35/50) + (20/110) + (10/150) + (x/250) <= 1

0,7 + 0,18 + 0,06 + (x/250) <= 1

0,94 + (x/250) <= 1

x/250 <= 1 - 0,94

x/250 <= 0,06

x <= 0,06*250

x <= 15

b) Para um processo P2 com Períodos de Eventos = 110, 250, 350, e tempo CPU = 50, 30, 100. Acrescentar um 4o evento cujo período seja de 1 segundo. Qual é o valor de X para que o sistema seja escalonável.

(50/110) + (30/250) + (100/350) + (x/1000) <= 1

0,45 + 0,12 + 0,28 + (x/1000) <= 1

0,85 + (x/1000) <= 1

x/1000 <= 1 - 0,85

x/1000 <= 0,15

x <= 0,15*1000

x <= 150

Concorrência entre Processos

15. a) Quais os valores corretos esperados para os saldos dos clientes A e B após o término da execução dos processos?

Cliente A = 200 e Cliente B = 1.200

b) Quais os valores finais dos saldos dos clientes se a sequência temporal de execução das operações for: 1a, 2a, 1b, 2b, 1c, 2c, 1d, 2d, 1e, 2e, 1f, 2f?

Cliente A = 400 e Cliente B = 1.100

c) Utilizando semáforos, proponha uma solução que garanta a integridade dos saldos e permita o maior compartilhamento possível dos recursos entre os processos, não esquecendo a especificação da inicialização dos semáforos.

Processo 1 (Cliente A)	Processo 2 (Cliente B)

/* saque em A */
Down (S1)
x := saldo_do_cliente_A;
x := x - 200;
saldo_do_cliente_A := x;
Up (S1)
	/*saque em A */
	Down (S1)
	y := saldo_do_cliente_A;
	y := y - 100;
	saldo_do_cliente_A := y;
	Up (S1)

/* deposito em B */
Down (S2)
x := saldo_do_cliente_B;
x := x + 100;
saldo_do_cliente_B := x;
Up (S2)
	/* deposito em B */
	Down (S2)
	y := saldo_do_cliente_B;
	y := y + 200;
	saldo_do_cliente_B := y;
	Up (S2)

16.

a) Suponha que exista apenas um leitor fazendo acesso à base. Enquanto este processo realiza aleitura, quais os valores das três variáveis?

Acesso=0 Exclusao=1 Nleitores=1

b) Chega um escritor enquanto o leitor ainda está lendo. Quais os valores das três variáveis após obloqueio do escritor ? Sobre qual(is) semáforo(s) se dá o bloqueio?

Acesso=0 Exclusao=1 Nleitores=1, o bloqueio ocorre no semáforo Acesso.

c) Chega mais um leitor enquanto o primeiro ainda não acabou de ler e o escritor está bloqueado.Descreva os valores das três variáveis quando o segundo leitor inicia a leitura.

Acesso=0 Exclusao=1 Nleitores=2

d) Os dois leitores terminam simultaneamente a leitura. É possível haver problemas quanto àintegridade do valor da variável nleitores? Justifique.

Não, pois a exclusão mútua a esta variável é implementada pelo semáforo Exclusao.

e) Descreva o que acontece com o escritor quando os dois leitores terminam suas leituras. Descrevaos valores das três variáveis quando o escritor inicia a escrita.

O processo Escritor inicia a escrita. Acesso=0 Exclusao=1 Nleitores=0

f) Enquanto o escritor está atualizando a base, chegam mais um escritor e mais um leitor. Sobrequal(is) semáforo(s) eles ficam bloqueados? Descreva os valores das três variáveis após obloqueio dos recém-chegados.

Os processo ficam bloqueados no semáforo Acesso. Acesso=0 Exclusao=0 Nleitores=1

g) Quando o escritor houver terminado a atualização, é possível prever qual dos processosbloqueados (leitor ou escritor) terá acesso primeiro à base?

Não, em geral os sistemas operacionais utilizam a escolha randômica dentre os processos em estado de espera.

h) Descreva uma situação onde os escritores sofram starvation (adiamento indefinido).

Caso um processo Escritor esteja aguardando, bloqueado pelo semáforo Acesso, e sempre surgirem novos processos Leitor, o processo Escritor pode nunca ganhar acesso ao recurso.