1) Em um sistema que suporta programação concorrente apenas através da troca de mensagens, será criado um Servidor para controlar o uso das portas seriais. Quando um processo Cliente deseja usar uma porta serial, ele envia uma mensagem “Aloca” para o Servidor. Existem N portas seriais, todas equivalentes, mas cada uma pode ser usada somente por um Cliente de cada vez. O Servidor informa ao Cliente a porta que ele vai usar através da mensagem “Porta p”. Ao concluir o uso, o Cliente envia para o Servidor a mensagem “Libera p”. Suponha que exista mais do que N processos Clientes. Mostre o algoritmo do Servidor, em português estruturado. Apresentar a solução como foi feito para o Servidor de Buffer visto em aula. Supor “receive” bloqueante.

2) O problema dos leitores/escritores consiste de um texto que pode ser lido ou escrito por vários processos. Considerando o código abaixo, responda justificando:

a) É possível vários leitores lerem ao mesmo tempo ?
b) É possível vários escritores escreverem ao mesmo tempo ?
c) É possível postergação indefinida de um leitor ?
d) É possível postergação indefinida de um escritor ?

```
int nl = 0;
semaphore tipo = 1;
semaphore exclusivo = 1;
```

```
void leitor( void)
{         ...
       P( exclusivo );
       if( nl > 0 )     ++ nl;
       else{
                        P( tipo );
                        nl = 1;
                        }
        V( exclusivo );
        Acessa o texto
        P( exclusivo );
         -- nl;
        if( nl == 0 )  V( tipo );
        V( exclusivo );
         ...
}
```

```
void escritor( void)
{         ...
        P( tipo );
        Acessa o texto
        V( tipo );
          ...
}
```

3) Em um programa concorrente existem N processos trabalhadores e um único processo finalizador. O programa é tal que o processo finalizador precisa esperar todos os processos trabalhadores terminarem para então ele finalizar o programa. Este tipo de sincronização é chamada de “barreira” e é típica da programação em máquinas paralelas.

Implemente uma solução para o problema da barreira usando mutexes e variáveis condição da biblioteca das pthreads. A solução consiste de 2 rotinas:

```
/* processo trabalhador “meuPid” informa que acabou sua parte do serviço */
        void  acabei (int meuPid);

/* processo finalizador fica bloqueado até todos os trabalhadores acabarem o serviço */
        void espera_todos( void );
```

Dicas sobre o Posix:

```
pthread_mutex_t  m = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock( &m );
pthread_mutex_unlock( &m );

pthread_cond_t vc = PTHREAD_COND_INITIALIZER;
pthread_cond_wait( &vc , &m );
pthread_cond_signal( &vc );
```

4) Determine se o conjunto de tarefas abaixo é escalonável com deadline monotônico, aplicando a análise para D ≤ P, vista em aula.

T1:	J1=0	C1=2	P1=5	D1=5	B1 = 0
T2:	J2=0	C2=5	P2=30	D2=10	B2 = 0
T3:	J3=0	C3=10	P3=25	D3=25	B3 = 0

$$W_i = C_i + B_i + \sum_{j \in HP(i)} \left\lceil \frac{J_j + W_i}{P_j} \right\rceil \times C_j \qquad R_i = J_i + W_i$$

5) Determine se o conjunto de tarefas abaixo é escalonável, aplicando a análise baseada em tempo de resposta vista em aula, quando:

(a) prioridades são atribuídas conforme o deadline monotônico;
(b) prioridades são atribuídas conforme o taxa monotônico (rate monotonic);

T1:	C1=3	P1=20	D1=6
T2:	C2=3	P2=15	D2=7
T3:	C3=4	P3=9	D3=8

6) Considerando o mesmo conjunto de tarefas, é possível criar uma escala de tempo não preemptiva para atender os deadlines solicitados ? Justifique tentando criar uma escala do tipo usada por executivo cíclico.

7) Ilustre através de exemplos como as características abaixo do sistema operacional podem aumentar o tempo de resposta de uma tarefa com alta prioridade. Para cada exemplo, desenhe um diagrama de tempo para mostrar o impacto sobre a tarefa de alta prioridade.

(a) Interrupções desabilitadas.
(b) Kernel não é preemptivo.

8) Indique dois aspectos do projeto de um sistema operacional que afetam o comportamento temporal de uma aplicação, atrapalhando o atendimento dos requisitos temporais.

9) Eventos são coletados em uma rede composta por 4 computadores, e associados com a hora local do computador que detecta cada evento. Para que os mesmos possam ser analisados em conjunto, é necessário um erro máximo entre relógios de 100 milisegundos. Os cristais usados nesses computadores possuem uma precisão de 10^{-5} em relação a sua freqüência nominal, para mais ou para menos.

Supondo que pode existir um erro inicial de 40 milissegundos entre quaisquer dois relógios da rede, calcule quanto tempo vai levar até que os dois relógios com maior erro entre eles apresentem um erro de 100 milisegundos, supondo sempre o pior caso.

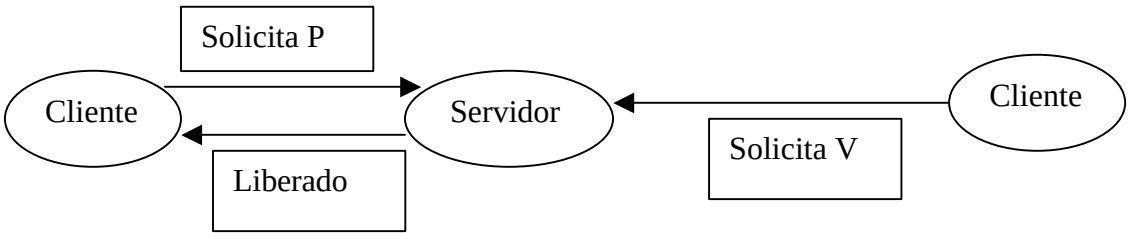
10) Desenhe o grafo de precedência do código abaixo:

```
main()
{
[1]     int f1, f2, f3;   /* Identifica processos filho*/

[2]     printf("Alo do pai\n");
[3]     f1 = fork( codigo_do_filho );          /* Cria filho 1 */
[4]     printf("Filho 1 criado\n");
[5]     f2 = fork( codigo_do_filho );          /* Cria filho 2 */
[6]     printf("Filho 2 criado\n");
[7]     wait( f1);
[8]     printf("Filho 1 morreu\n");
[9]     f3 = fork( codigo_do_filho );          /* Cria filho 3 */
[10]    printf("Filho 3 criado\n");
[11]    wait( f3);
[12]    printf("Filho 3 morreu\n");
[13]    wait( f2);
[14]    printf("Filho 2 morreu\n");
[15]    exit();
}

codigo_do_filho()
{
[16]    printf("Alo do filho\n");
[17]    exit();
}
```

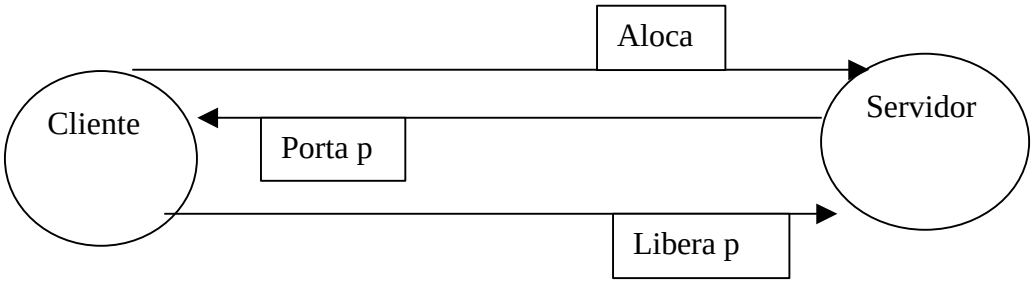
11) Em um sistema que suporta programação concorrente apenas através da troca de mensagens, será criado um Servidor de Semáforo. Suponha que o processo S será o servidor, implementando a funcionalidade de um único semáforo inicializado com 1. O protocolo que permite a um processo cliente executar operações P e V é mostrado abaixo. Mostre o algoritmo do processo S, em português estruturado. Apresentar a solução como foi feito para o Servidor de Buffer visto em aula. Supor “receive” bloqueante.

P: --valor
Se valor < 0 então bloqueia quem executou o P

V: ++valor
Acorda o primeiro da fila, se houver alguém bloqueado



12) Em um sistema que suporta programação concorrente apenas através da troca de mensagens, será criado um Servidor para controlar o uso das portas seriais. Quando um processo Cliente deseja usar uma porta serial, ele envia uma mensagem “Aloca” para o Servidor. Existem N portas seriais, todas equivalentes, mas cada uma pode ser usada somente por um Cliente de cada vez. O Servidor informa ao Cliente a porta que ele vai usar através da mensagem “Porta p”. Ao concluir o uso, o Cliente envia para o Servidor a mensagem “Libera p”. Suponha que exista mais do que N processos Clientes. Mostre o algoritmo do Servidor, em português estruturado. Apresentar a solução como foi feito para o Servidor de Buffer visto em aula. Supor “receive” bloqueante.



13) Suponha a existência de um texto que é acessado por vários processos Leitores e um processo Escritor. Quando o Escritor está acessando, nenhum Leitor pode ler o texto, pois poderia acessar uma versão inconsistente. Por outro lado, é permitido que vários Leitores acessem o texto simultaneamente. Implemente a solução descrita abaixo, usando os recursos da biblioteca Pthreads. Não é necessário criar as threads, apenas mostrar o código de acesso ao texto.

Leitor:
```
Se número de leitores acessando for maior que zero
        Então  Incrementa número de leitores acessando
        Senão  Se escritor acessando
                        Então  Espera escritor sair
                        Senão  Faz Leitores acessando igual a 1
Acessa o texto
Decrementa número de leitores acessando
Se número de leitores acessando for igual a zero
        Então  Libera o texto
```

Escritor:
```
Se número de leitores acessando for maior que zero
        Então  Espera ficar livre
        Senão  Indica escritor acessando
Acessa o texto
Libera o texto
```

Dicas sobre o Posix:

```
pthread_mutex_t  m = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock( &m );
pthread_mutex_unlock( &m );

pthread_cond_t vc = PTHREAD_COND_INITIALIZER;
pthread_cond_wait( &vc , &m );
pthread_cond_signal( &vc );
```

14) Considerando o código abaixo, uma variação da implementação do produtor/consumidor vista em aula, responda as seguintes perguntas (justificando):
a) Está garantido o acesso exclusivo à variável "buffer" no caso de 1 produtor e 1 consumidor ?
b) É possível a ocorrência de deadlock no caso de 2 produtores e 2 consumidores ?

```
struct  tipo_dado  buffer;
semaphore espera_vaga = 1;
semaphore espera_dado = 0;
```

void produtor(void)
{ ...
P(espera_vaga);
buffer = dado_produzido;
V(espera_dado);
...
}
	void consumidor(void)
	{ ...
	P(espera_dado);
	dado_a_consumir = buffer;
	V(espera_vaga);
	...
	}

15) Descreva a diferença entre deadlines hard, firm e soft, e indique um exemplo para cada um deles.

16) Usando os mecanismos "mutex" e "variáveis condição" do Posix, implemente as operações P e V de um semáforo. Cada uma dessas operações deverá ser substituida por um código C com semântica similar. Lembre-se que estas operações devem ser atômicas. O valor inicial do semáforo é 1.

Dicas sobre o Posix:

```
pthread_mutex_t  m = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock( &m );
pthread_mutex_unlock( &m );

pthread_cond_t vc = PTHREAD_COND_INITIALIZER;
pthread_cond_wait( &vc , &m );
pthread_cond_signal( &vc );
```

17) Considerando o código abaixo, uma variação da implementação do produtor/consumidor vista em aula, responda as seguintes perguntas (justificando):
a) Está garantido o acesso exclusivo à variável "buffer" no caso de 1 produtor e 1 consumidor ?
b) É possível a ocorrência de deadlock no caso de 2 produtores e 2 consumidores ?

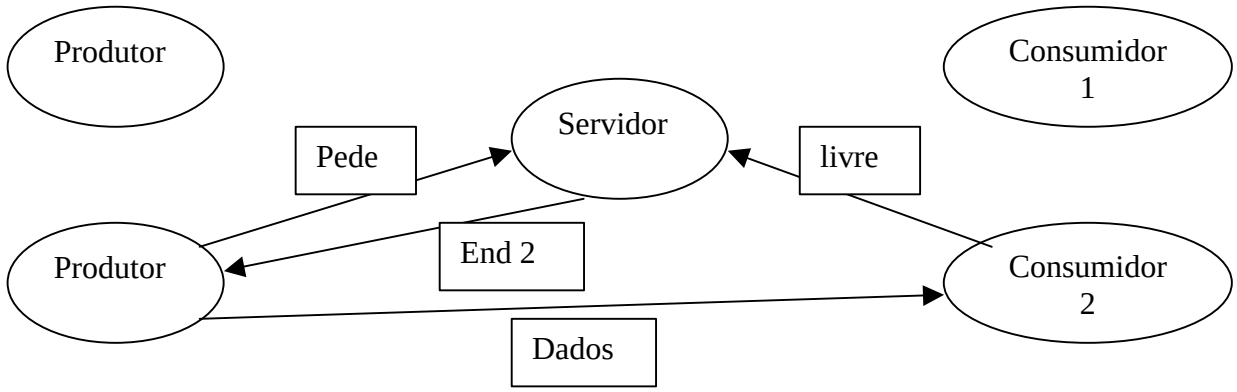
```
struct  tipo_dado  buffer;
semaphore espera_vaga = 1;
semaphore espera_dado = 0;
```

void produtor(void)
{ ...
P(espera_vaga);
buffer = dado_produzido;
V(espera_dado);
...
}
	void consumidor(void)
	{ ...
	P(espera_dado);
	dado_a_consumir = buffer;
	V(espera_vaga);
	...
	}

18) Indique dois aspectos do projeto de um sistema operacional que afeta o comportamento temporal de uma aplicação.

19) Qual a diferença entre tempo real crítico e tempo real brando ? Forneça também uma aplicação exemplo para cada tipo.

20) Em um sistema que suporta programação concorrente apenas através da troca de mensagens, será criado um Servidor para controlar o envio de dados entre produtores e consumidores. Os dados serão enviados diretamente de um Produtor para um Consumidor livre, mas o Produtor deve consultar o Servidor para obter o número do Consumidor que ele deverá usar. O Servidor deve manter uma lista com os números dos Consumidores livres no momento, e informar um desses números quando um Produtor solicita. Após receber e processar os dados, o Consumidor volta a informar que está livre para o Servidor. Mostre o algoritmo do Servidor, em português estruturado. Apresentar a solução como foi feito para o Servidor de Buffer visto em aula. Supor "receive" bloqueante.



21) O problema dos leitores/escritores consiste de um texto que pode ser lido ou escrito por vários processos. Considerando o código abaixo, responda justificando:

e) É possível vários leitores lerem ao mesmo tempo ?
f) É possível vários escritores escreverem ao mesmo tempo ?
g) É possível um leitor e um escritor acessarem ao mesmo tempo ?
h) É possível postergação indefinida de um escritor ?

```
int nl = 0;
semaphore tipo = 1;
semaphore exclusivo = 1;
```

```
void leitor( void)
{       ...
      P( exclusivo );
      if( nl > 0 )    ++ nl;
      else{
                  P( tipo );
                  nl = 1;
                  }
       V( exclusivo );
       Acessa o texto
       P( exclusivo );
        -- nl;
       if( nl == 0 )  V( tipo );
       V( exclusivo );
        ...
}
```

```
void escritor( void)
{       ...
      P( exclusivo );
      Acessa o texto
      V( exclusivo );
        ...
}
```

22) Usando **semáforos**, implemente rotinas cujo comportamento seja similar ao das funções wait e signal vistas em aula (pthreads) e descritas abaixo. Crie tantos semáforos e variáveis auxiliares do tipo inteiro quantos julgar necessário. Explique a lógica da solução.

WAIT(VC): Sempre bloqueia o processo até alguém executar um SIGNAL.

SIGNAL(VC): Se existe alguém bloqueado em VC, acorda apenas 1 deles.

Obs: O que é pedido são dois trechos de código em C, tal que um funcione como se fosse o "wait" e o outro funcione como se fosse o "signal". Neste código em C, para obter atomicidade e bloquear tarefas, devem ser usados semáforos.

23) Determine se o conjunto de tarefas abaixo é escalonável se prioridades forem atribuidas segundo o deadline monotônico, aplicando a análise para D ≤ P, vista em aula. Considere que existe uma seção crítica entre T1 e T2, em função de uma tabela acessada pelas duas tarefas. A tarefa T1 permanece 1 unidade de tempo acessando a tabela, enquanto a tarefa T2 permanece 2 unidades de tempo acessando a mesma tabela.

T1:	J1=0	C1=2	P1=5	D1=5
T2:	J2=0	C2=5	P2=30	D2=10
T3:	J3=0	C3=10	P3=25	D3=25

$$W_i = C_i + B_i + \sum_{j \in HP(i)} \left\lceil \frac{J_j + W_i}{P_j} \right\rceil \times C_j \qquad R_i = J_i + W_i$$

24) Considerando o mesmo conjunto de tarefas, é possível criar uma escala de tempo não preemptiva para atender os deadlines solicitados ? Justifique tentando criar uma escala do tipo usada por executivo cíclico.

25) Considere um kernel preemptivo com as seguintes características:

- Interrupções do timer acontecem a cada 10 ms;
- O tratador de interrupções do timer demora 1 ms para executar;
- Dentro do kernel as interrupções ficam desabilitadas no máximo por 2 ms de cada vez;
- A carga do contexto de uma tarefa demora 0,1 ms e o salvamento do contexto de uma tarefa demora 0,2 ms.

Supondo que a tarefa da aplicação de mais alta prioridade execute a cada ativação um algoritmo que demora (tempo de computação) no pior caso 200 ms, qual será o seu tempo máximo de resposta ? Justifique cada termo da equação usada no cálculo.