1. (recurso 2013/2014) Considere o seguinte programa:

```
1:  int i = 0;
2:
3:  void *mythread(void *arg) {
4:    int *p = (int *) arg;
5:    int j;
6:    sem_wait(sem1);
7:    i = i + 1;
8:    p[0] = 10;
9:    j = j + 1;
10:   printf("%d: %d\n", getpid(), i);
11:   return NULL
12: }
13:
14: int main() {
15:   pthread_t tid;
16:   int j = 0;
17:
18:   sem_t *sem1 = sem_open("sem1",
19:    O_CREAT | O_RDWR, 0666, 0);
20:   pthread_create(&tid, NULL, mythread, &j);
21:
22:   i = i + 1;
23:   sem_post(sem1);
24:   pthread_join(tid, NULL);
25:   j = j + 1;
26:   printf("%d: %d %d\n", getpid(), i, j);
27:   return 0;
28: }
29:
30:
```

Indique uma possível sucessão de mensagens impressas no ecrã após a execução do programa. Assuma que não existem interferências de outros processos no sistema, que o identificador do processo inicial é 2000 e que o(s) novo(s) processo(s) toma(m) o(s) valor(es) seguinte(s). Justifique.

2. (época normal 2013/14) Analise o seguinte extrato de um programa:

```
1:  #define N 4
2:
3:  int c;
4:
5:  typedef struct { int t; double *dados; int n;} args_t;
6:
7:  void *processa(void *arg) {
8:    args_t *p = (args_t *) arg;
9:    c++;
10:   processa2(p->dados, p->n);//operação muito demorada
11:   printf("t: %d\n", p->t);
12:   return NULL;
13: }
14:
15: int main() {
16:   int i;
17:   pthread_t tid[N];
18:   double dados[N*1024];
19:   args_t args;
20:
21:   c = 0;
22:   for(i=0;i<N;++i) {
23:     args.t = i; args.dados = dados+i*1024; args.n = 1024;
24:     pthread_create(tid+i, NULL, processa, &args);
25:   }
26:
27:   for(i=0;i<N;++i)
28:     pthread_join(tid[i], NULL);
29:
30:   printf("c = %d\n", c);
31: }
```

a) Apresente a sequência de mensagens impressas por este programa. Justifique sucintamente.

b) Justifique a necessidade de incluir as as linhas 27 e 28.

c) Assumindo que o tempo de execução deste programa num processador *quad-core* (4 núcleos de processamento) é de T segundos, indique, justificando, qual o tempo de execução expectável do mesmo nos seguintes cenários (mantendo N=4 em todos os casos): a) num processador *dual-core* (dois núcleos de processamento); b) num processador *octo-core* (8 núcleos de processamento). Assuma que todos os núcleos de processamento têm as mesmas características e que não existem outras tarefas a competir pela utilização dos processadores.

3. (recurso 2014/2015) Analise o seguinte extrato de um programa:

```
1:  sem_t sem;
2:  int h = 0;
3:
4:  typedef struct { int i; int *c;} args_t;
5:
6:  void *t1(void *arg) {
7:    args_t *p = (args_t *) arg;
8:    sem_wait(&sem);
9:    ++h;
10:   sleep(2);
11:   printf("t1: %d %d %d\n", p->i, p->c[0], h);
12:   sleep(2);
13:   p->c[0] += p->i;
14:   sleep(p->i);
15:   printf("Fim: %d\n", p->c[0]);
16: }
17:
18: int main() {
19:   pthread_t tid1, tid2;
20:   args_t args1, args2;
21:   int k = 20;
22:
23:   sem_init(&sem, 0, 1);//valor inicial 1
24:
25:   args1.i = 20;  args1.c = &k;
26:   pthread_create(&tid1, NULL, t1, &args1);
27:
28:   args2.i = 10;  args2.c = &k;
29:   pthread_create(&tid2, NULL, t1, &args2);
30:
31:   sem_post(&sem);
32:   pthread_join(tid2, NULL);
33:
34:   printf("main: %d %d\n", k, h);
35: }
```

Apresente a sequência de impressões produzidas por este programa, assumindo a sua execução num computador típico. Apresente um diagrama temporal representativo da execução do programa e justifique sucintamente. A sequência de impressões deve ser apresentadas de forma destacada.

4. (época normal 2014/2015) Analise o seguinte extrato de um programa:

```
1:  sem_t sem;
2:
3:  typedef struct { int i; char c;} args_t;
4:
5:  void *t1(void *arg) {
6:    args_t *p1 = (args_t *) arg;
7:    sem_wait(&sem);
8:    printf("t1: %d %c\n", p1->i, p1->c);
9:  }
10:
11: void *t2(void *arg) {
12:   args_t *p2 = (args_t *) arg;
13:   printf("t2: %d %c\n", p2->i, p2->c);
14:   p2->i = 40;
15:   sem_post(&sem);
16:   sleep(4);
17:   printf("t2: %d %c\n", p2->i, p2->c);
18: }
19:
20: int main() {
21:   pthread_t tid1, tid2;
22:   args_t args;
23:
24:   sem_init(&sem, 0, 0);//valor inicial 0
25:
26:   args.i = 20;  args.c = 'a';
27:   pthread_create(&tid1, NULL, t1, &args);
28:
29:   args.i = 10;  args.c = 'b';
30:   pthread_create(&tid2, NULL, t2, &args);
31:   pthread_join(tid1, NULL);
32:
33:   printf("main: %d %c\n", args.i, args.c);
34: }
```

Apresente a sequência de impressões produzidas por este programa, assumindo a sua execução num computador típico. Apresente um diagrama temporal representativo da execução do programa e justifique sucintamente. A sequência de impressões deve ser apresentadas de forma destacada.

Sugestões de solução

1. Este programa cria uma nova thread na linha 22 (pthread_create). A variável j da função main vai ser alterada indiretamente através da variável p da função mythread.

Thread principal	**Nova thread**
`pthread_create`	`mythread(&j)`
`i ← 1`	`sem_wait (aguarda, pois o semáforo está a 0)`
`sem_post =>`	…
`pthread_join`	`i ← 2`
…	`p[0] (variável j da main) ← 10`
…	`printf`
…	`(termina thread)`
`j ← 11`	
`(termina processo)`	

```
2000: 2
2000: 2 11
```

2. a+b) A sucessão de mensagens seria:

t: 3
t: 3
t: 3
t: 3
c = 4

Todas as *threads* recebem um apontador para a mesma variável args da função main (4º parâmetro da função pthread_create na linha 28). Como a função processa2 é mais demorada do que o lançamento das 4 *threads*, quando cada *thread* chega à linha 11, vão imprimir o último valor escrito no campo t (linha 24), apesar de este ser diferente na altura da criação de cada *thread*.
(b) O objetivo das linhas 27 e 28 é esperar que todas as *threads* terminem antes que o processo termine. Caso não fossem incluídas, o processo terminaria imediatamente e, consequentemente, também todas as suas *threads*, impedindo a execução da função processa2.
Como a variável c é global, vai ficar com o resultado acumulado da execução da linha 9 em cada uma das *threads*.

c) O tempo de execução num processador dual-core será de 2T, uma vez que o sistema apenas conseguirá processar 2 *threads* em paralelo, em oposição às 4 do caso inicial. No processador octo-core o tempo de execução será T, uma vez que, apesar de estarem disponíveis 8 núcleos de processamento, o programa só solicita a execução simultânea de 4 *threads*.

3. Possível execução:

h = 0 **sem = 1**		
pthread_create pthread_create	nova *thread* executa t1(&args1)	nova *thread* executa t1(&args2)
sem_post => **sem = sem+1 = 2** pthread_join(tid2,...)	sem_wait() => **sem=sem-1=1** ++h => h = 1 sleep(2) . . . printf (linha 11) sleep(2) . . . (linha 13) k = k + args1.i <=> k = 20 + 20 = 40 sleep(20)	sem_wait() => **sem=sem-1=0** ++h => h = 2 sleep(2) . . . printf (linha 11) sleep(2) . . . (linha 13) k = k + args2.i <=> k = 40 + 10 = 50 sleep(10) . . . printf (linha 15)
printf (linha 34)	.	(termina a thread)
(processo termina)	(não chega a fazer a impressão da linha 15)	

t1: 20 20 2
t1: 10 20 2
Fim: 50
main: 50 2

4. A variável args estará acessível em ambas as *threads* criadas. A primeira *thread*, que tem t1 como função principal, acede a args através de p1 (linhas 27 e 6). A segunda *thread*, que tem t2 como função principal, acede a args através de p2 (linhas 30 e 12).

```
sem = 0 (linha 24)
args.i = 20
args.c = 'a'
pthread_create (linha 27)      nova thread, executa t1
args.i = 10                    sem_wait (linha 7) – semáforo a
args.c = 'b'                   0, aguarda.
pthread_create (linha 30)                                  nova thread, executa t2
pthread_join(tid1)   –   aguarda                           impressão "t2: 10 b"
que a primeira thread termine.                             args.i = 40

                               (desbloqueia)               sem_post
                               impressão "t1: 40 b"        espera durante 4 segundos
                               termina thread
pthread_join retorna                                       Nota: esta thread não chega a
impressão: "main: 40 b"                                    terminar a sua execução.
processo termina
```

t2: 10 b
t1: 40 b
main: 40 b