MC514
Sistemas Operacionais:
Teoria e Prática

1s2006

Dicas para a resolução da Lista 1

Questão 9

Algoritmo da padaria

- Baseado em espera ocupada
- Permite que várias threads se organizem para pegar senhas e entrar na região crítica
- Como tentar simular *exatamente* o mesmo comportamento com locks e variáveis de condição?

Questão 9 - Algoritmo da padaria

Primeira tentativa

```
num[N] = { 0, 0, ..., 0 }
```

Thread_i:

```
num[i] = max (num[0]...num[N-1]) + 1

for (j = 0; j < N; j++)
  while (num[j] != 0 &&  num[j] < num[i]) ;

s = i;
print ("Thr ", i, s);

num[i] = 0;
```

Questão 9 - Algoritmo da padaria

Segunda tentativa

```
num[N] = { 0, 0, ..., 0 }
```

Thread_i:

```
num[i] = max (num[0]...num[N-1]) + 1

for (j = 0; j < N; j++)
  while (num[j] != 0 &&
          (num[j] < num[i] || num[i] == num[j] && j < i));

s = i;
print ("Thr ", i, s);

num[i] = 0;
```

Questão 9 - Algoritmo da padaria

```
escolhendo[N] = { false, false, ..., false }
num[N] = { 0, 0, ..., 0 }
```

Thread_i:

```
escolhendo[i] = true;
num[i] = max (num[0]...num[N-1]) + 1
escolhendo[i] = false;
for (j = 0; j < N; j++)
  while (escolhendo[j]) ;
  while (num[j] != 0 &&
          (num[j] < num[i] || num[i] == num[j] && j < i));
s = i;
print ("Thr ", i, s);
num[i] = 0;
```

Questão 9 - Algoritmo da padaria

- Primeira tentativa com um único lock e uma única variável de condição.
- Quais são as diferenças de comportamento?

Questão 9 - Algoritmo da padaria

```
lock(mutex);
num[i] = max (num[0]...num[N-1]) + 1;
for (j = 0; j < N; j++)
  while (num[j] != 0 &&
        (num[j] < num[i] || num[i] == num[j] && j < i))
           cond_wait(&wait, &mutex);
unlock(mutex);
s = i;
print ("Thr ", i, s);
lock(mutex);
num[i] = 0;
cond_broadcast(wait);
unlock(mutex);
```

Questão 9 - Algoritmo da padaria

- Com um único lock e uma única variável de condição as threads nunca pegam senhas repetidas.
- Como aumentar o paralelismo?

Questão 9 - Algoritmo da padaria

```
lock(mutex[i]);
num[i] = max (num[0]...num[N-1]) + 1;
unlock(mutex[i]);

for (j = 0; j < N; j++) {
  lock(mutex[j]);
  while (num[j] != 0 &&
        (num[j] < num[i] || num[i] == num[j] && j < i))
           cond_wait(&wait[j], &mutex[j]);
  unlock(mutex[j]);
}
```

Questão 9 - Algoritmo da padaria

```
s = i;
print ("Thr ", i, s);

lock(mutex[i]);
num[i] = 0;
cond_broadcast(wait[i]);
unlock(mutex[i]);
```

Questão 9 - Algoritmo da padaria

- O algoritmo anterior aparentemente funciona, mas os valores da senha são lidos sem um lock.
- Esta abordagem funcionaria em todas as máquinas? E se a senha fosse calculada sobre uma estrutura composta por mais de um campo?
- O que fazer para contornar este problema?

Questão 9 - Algoritmo da padaria

```
lock(mutex_esc[i]);
esc[i] = true;
unlock(mutex_esc[i]);

aux = max_com_locks(num[0]...num[N-1]) + 1;

lock(mutex[i]);
mutex[i] = aux;
unlock(mutex[i]);

lock(mutex_esc[i]);
esc[i] = false;
cond_broadcast(cond_esc[i]);
unlock(mutex_esc[i]);
```

Questão 9 - Algoritmo da padaria

```
for (j = 0; j < N; j++) {
  lock(mutex_esc[j]);
  if (esc[j])
      cond_wait(&wait_esc[j], &mutex_esc[j]);
  unlock(mutex_esc[j]);

  lock(mutex[j]);
  while (num[j] != 0 &&
        (num[j] < num[i] || num[i] == num[j] && j < i))
           cond_wait(&wait[j], &mutex[j]);
  unlock(mutex[j]);
}
```

Questão 9 - Algoritmo da padaria

```
s = i;
print ("Thr ", i, s);

lock(mutex[i]);
num[i] = 0;
cond_broadcast(wait[i]);
unlock(mutex[i]);
```