MC514–Sistemas Operacionais: Teoria e Prática

1s2008

Processos e Threads 6

Objetivos

- Abordagem da Thread Gerente
- Algoritmo da Padaria
- Prioridades para Threads

Algoritmos de Exclusão Mútua

- E se tivéssemos uma thread gerente?
  - Os algorimos seriam mais simples?
  - Qual o grande ponto negativo desta abordagem?
- Veja os programas: gerente?.c

Algoritmo da Padaria

- Análogo a um sistema de distribuição de senhas a clientes em uma loja
- A thread com a senha de menor número é atendida
- A própria thread deve escolher o seu número

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;
```

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;
```

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;
```

Black-White Bakery

Gadi Taubenfe

- The Black-White Bakery Algorithm. Proceedings of the 18th international symposium on distributed computing, Amsterdam, the Netherlands, October 2004. In: LNCS 3274 Springer Verlag 2004, 56-70
- rodadas de senhas coloridas
- permite senhas de tamanho fixo

Filas de prioridades diferentes

- Suponha que o gerente da padaria está pensando em implantar atendimento especial a idosos e gestantes
- Existem threads prioritárias e outras menos prioritárias;
- Nenhuma thread menos prioritária é atendida se houver uma thread mais prioritária esperando;
- Se uma thread menos prioritária estiver sendo atendida, a mais prioritária deve esperar;

Modificação para duas filas

Duas instâncias do algoritmo da padaria

```
#define N 10
#define M 5

esc[N] = { false, false, ..., false }
num[N] = { 0, 0, ..., 0 }

esc_pri[M] = { false, false, ..., false }
num_pri[M] = { 0, 0, ..., 0 }
```

Modificação para duas filas

Uma instância do algoritmo do desempate

```
#define PRI 0
#define NAO_PRI 1
int vez;
int interesse[2];
```

Thread menos prioritária

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

Thread mais prioritária (?)

```
esc_pri[i] = true;
num_pri[i] = max (num_pri[0]...num_pri[M-1]) + 1
esc_pri[i] = false;
for (j = 0; j < M; j++)
  while (esc_pri[j]) ;
  while (num_pri[j] != 0 && (num_pri[j] < num_pri[i] ||
         num_pri[i] == num_pri[j] && j < i));
interesse[PRI] = 1;
vez = PRI;
while (vez == PRI && interesse[NAO_PRI]);
s = i;
print ("Thr ", i, s);
interesse[PRI] = 0;
num_pri[i] = 0;
```

Thread mais prioritária

```
/* Código da padaria simples */
if (!interesse[PRI]){
  interesse[PRI] = 1;
  vez = PRI;
  while (vez == PRI && interesse[NAO_PRI]);
}
/* Região crítica */
if (não existe j!= i : num_pri[j] > 0)
  interesse[PRI] = 0;
num_pri[i] = 0;
```