

MC514–Sistemas Operacionais: Teoria e Prática
1s2010

Processos e Threads 6
Futexes e Padaria

Objetivos

- Futex
- Alternância
- Algoritmo da Padaria

Prototype

```
long sys_futex (  
    void *addr1,  
    int op,  
    int val1,  
    struct timespec *timeout,  
    void *addr2,  
    int val3);  
  
int syscall(SYS_futex, addr1, FUTEX_XXXX,  
            val1, timeout, addr2, val3);
```

FUTEX_WAIT

```
/* Retorna -1 se o futex não bloqueou e  
   0 caso contrário */  
int futex_wait(void *addr, int val1) {  
    return syscall(SYS_futex, addr, FUTEX_WAIT,  
                   val1, NULL, NULL, 0);}
```

- Bloqueio até notificação
- Não há bloqueio se `*addr1 != val1`
- Veja o código `ex0.c`

FUTEX_WAKE

```
/* Retorna o número de threads acordadas */  
int futex_wake(void *addr, int n) {  
    return syscall(SYS_futex, addr, FUTEX_WAKE,  
                   n, NULL, NULL, 0);}
```

- Quantas threads acordar?
 - 1
 - 5
 - INT_MAX (todas)
- Veja os códigos ex1.c e ex2.c

Abordagem da Alternância

```
int s = 0;  
int vez = 1; /* Primeiro a thread 1 */
```

Thread 0

```
while (true)  
    while (vez != 0);  
    s = 0;  
    print ("Thr 0:" , s);  
    vez = 1;
```

Thread 1

```
while (true)  
    while (vez != 1);  
    s = 1;  
    print ("Thr 1:" , s);  
    vez = 0;
```

- Modifique o código: `futex-alternancia.c`

Abordagem da Alternância

N threads

Thread_i:

```
while (true)
    while (vez != i);
    s = i;
    print ("Thr ", i, ": ", s);
    vez = (i + 1) % N;
```

- Modifique o código: futex-alternanciaN.c

Lock

- Atomic Builtins
- Veja o código: `futex-lock.c`

Algoritmo do Desempate (1981)

```
int s = 0, ultimo = 0, interesse[2] = {false, false};
```

Thread 0

```
while (true)
    interesse[0] = true;
    ultimo = 0;
    while (ultimo == 0 &&
           interesse[1]);
    s = 0;
    print ("Thr 0:" , s);
    interesse[0] = false;
```

Thread 1

```
while (true)
    interesse[1] = true;
    ultimo = 1;
    while (ultimo == 1 &&
           interesse[0]);
    s = 1;
    print ("Thr 1:" , s);
    interesse[1] = false;
```

- Modifique o código: futex-desempate.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;
```

Futex e padaria

- Como implementar padaria com futex?
- Quantos futexes são necessários?
- Trabalhe com o código futex-padaria.c

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;
- Como implementar?