Instituto de Informática - UFRGS

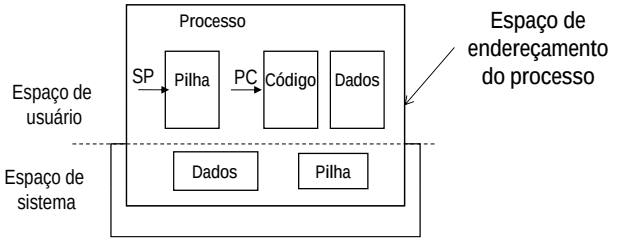
Sistemas Operacionais

Multithreading

Aula 05

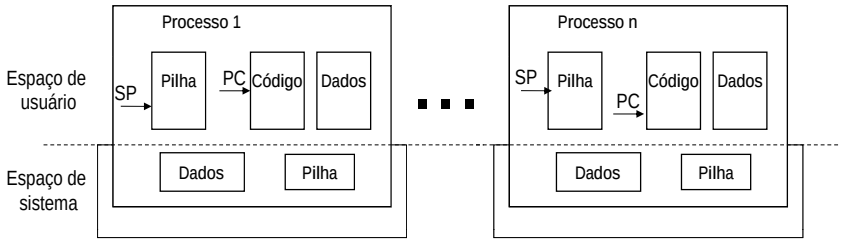
O modelo de processo (relembrando...)

- Processo é representado por:
  - Espaço de endereçamento: área p/ armazenamento da imagem do processo
  - Estruturas internos do sistema (tabelas, descritor de processos etc)
  - Contexto de execução: valores em memória (pilha, dados, heap) recursos alocados e registradores internos da CPU em um instante *t*
- Um fluxo de controle por processo (*thread*)

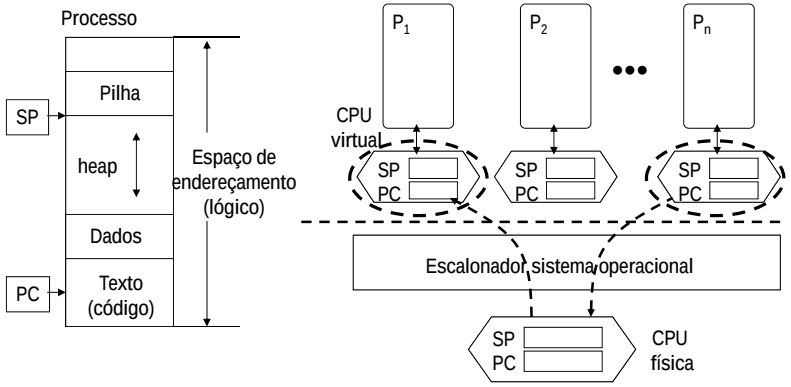


Processos e multiprogramação....

- Troca de processo implica em atualizar estruturas de dados internas do sistema operacional
  - e.g.; contexto, espaço de endereçamento, etc...
- Comunicação entre processos é via IPC



Multiprogramação com processo...



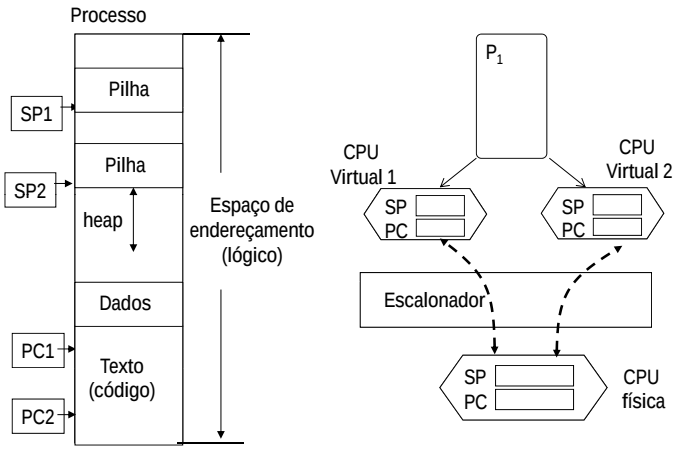
```
int fatorial(int *n) {
    int fat=1;

    for(;*n > 1; --*n) {
       fat = *n * fat;
    }
    return(fat);
}

void fibonnaci(int n){
    int fi, fj, fk, k;

    fi = 0;
    fj = 1;
    printf ("0-1");
    for ( k=1; k <= n; k++) {
        fk = fi + fj;
        fi = fj;
        fj = fk;
        printf("-%d", fk);
    }
    return;
}

int main(int argc, char * argv) {
    int n=5;

    fibonnaci(n);
    printf("\nFatorial: %d\n", fatorial(&n));
    exit(0);
}
```

PC →

SP →

End. retorno
5
fi
fj
fk
k

E se fosse feito algo assim ?




```
int fatorial(int *n) {
    int fat=1;

    for(;*n > 1; --*n) {
       fat = *n * fat;
    }
    return(fat);
}

void fibonnaci(int n){
    int fi, fj, fk, k;

    fi = 0;
    fj = 1;
    printf ("0-1");
    for ( k=1; k <= n; k++) {
        fk = fi + fj;
        fi = fj;
        fj = fk;
        printf("-%d", fk);
    }
    return;
}

int main(int argc, char * argv) {
    int n=5;

    fibonnaci(n);
    printf("\nFatorial: %d\n", fatorial(&n));
    exit(0);
}
```

PC2 →

PC1 →

SP2 →

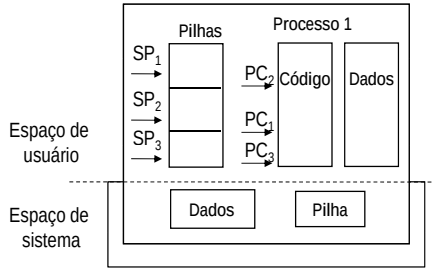
End. retorno
end. n
fat

SP1 →

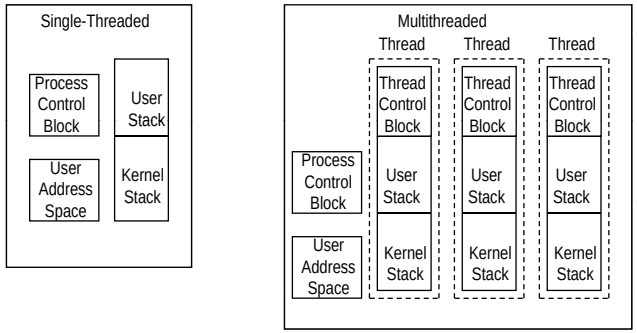
End. retorno
5
fi
fj
fk
k

Vários fluxos em um único processo: *multithreading*

- Um fluxo de instrução é implementado através do contador de programa (PC) e de uma pilha (SP)
- Thread(s) "vive(m)" dentro de um processo
- Estruturas comuns compartilhadas
  - Código
  - Dados
  - Descritor de processo
- Conceito de *multithreading*

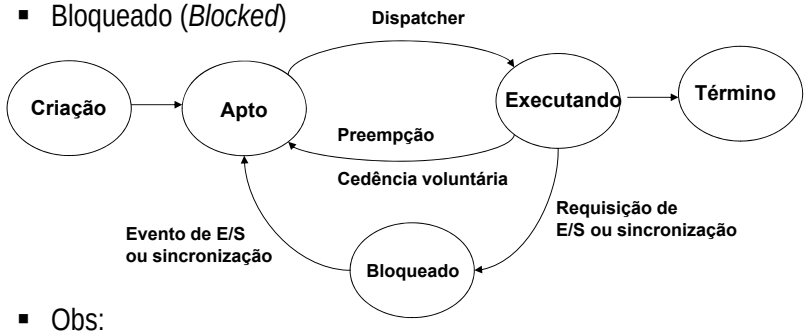


Modelos de processos *single Threaded* e *multithreaded*



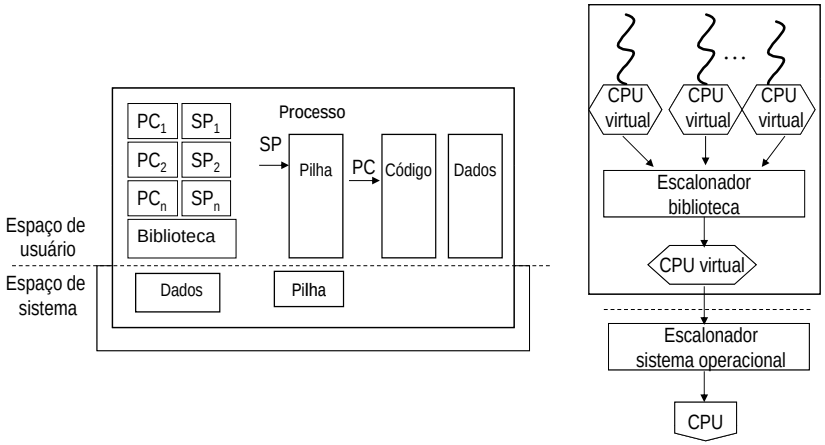
Estados de uma *thread*

- Apto (*Ready*)
- Executando (*Running*)
- Bloqueado (*Blocked*)

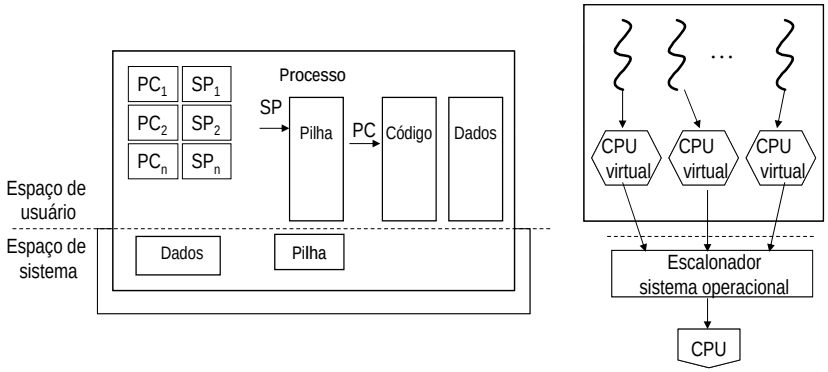


- Obs:
  - Assim como em processos, a preempçao por tempo esgotado pressupõe existência e uso de interrupção de tempo

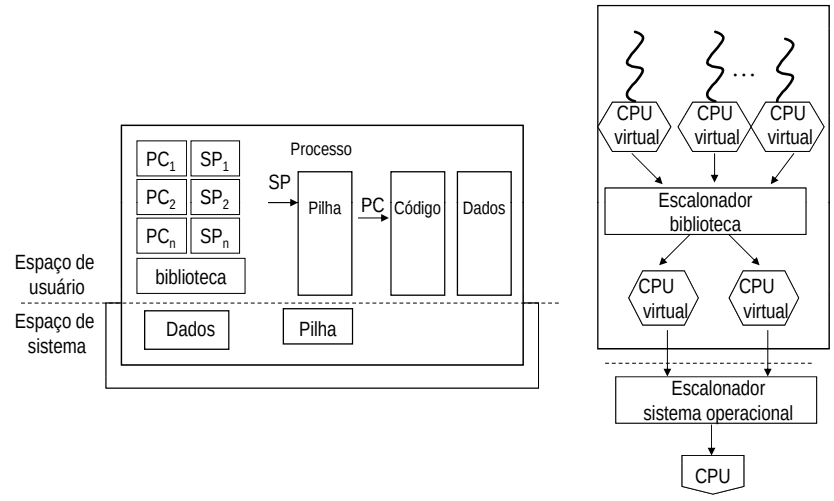
Implementação modelo N:1



Implementação modelo 1:1



Implementação modelo M:N



Exemplo de programação com *threads* (POSIX)

```
#include <pthread.h>
int g;

void do_it_1 (void *arg) {
        int i, n = *(int *)arg;
        for (i = 0; i < n; i++) g++;
}

void do_it_2 (void *arg) {
        int i, n = *(int *)arg;
        for (i = 0; i < n; i++)
        printf("%d\n", g);
}
```

```
int main( int argc, char **argv) {
        pthread_t th1, th2;
        int n = 10;
        pthread_create( &th1, NULL, do_it_1, &n);
        pthread_create( &th2, NULL, do_it_2, &n);

        pthread_join (th1, NULL);
        pthread_join (th2 , NULL);
        exit(0);
}
```


Leituras complementares

- A. Tanenbaum. Sistemas Operacionais Modernos (3ª edição), Pearson Brasil, 2010.
  - Capítulo 2: seções 2.2.1 a 2.2.5 e 2.2.9
- R. Oliveira, A. Carissimi, S. Toscani *Sistemas Operacionais* Editora Bookman 2010.
  - Capítulo 4: seção 4.3
- A. Silberchatz, P. Galvin, G. Gagne; *Sistemas Operacionais com Java* (7ª edição). Campus. 2008.
  - Capítulo 4