MC504/MC514 - Sistemas Operacionais

Processos e Threads: Introdução

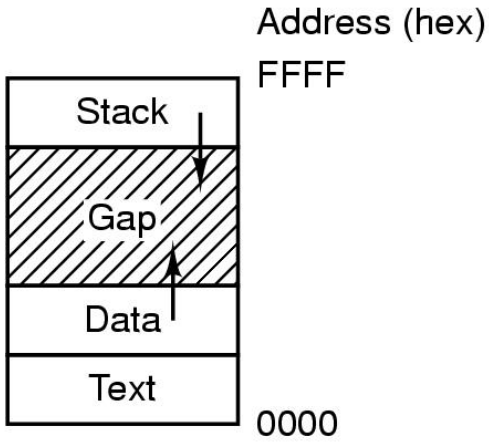
Islene Calciolari Garcia

Primeiro Semestre de 2016

Sumário

Processo

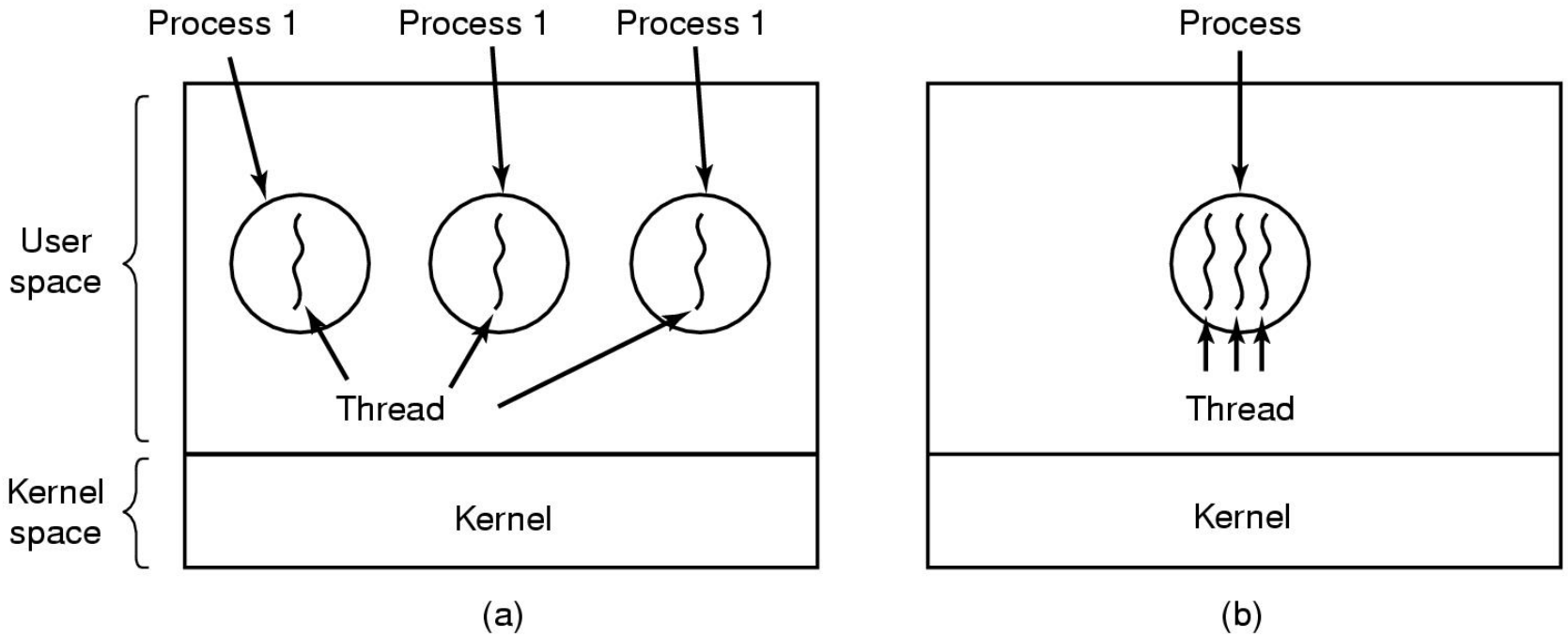
- Programa em execução
- Espaço de endereçamento



Tanenbaum: Figura 1.20

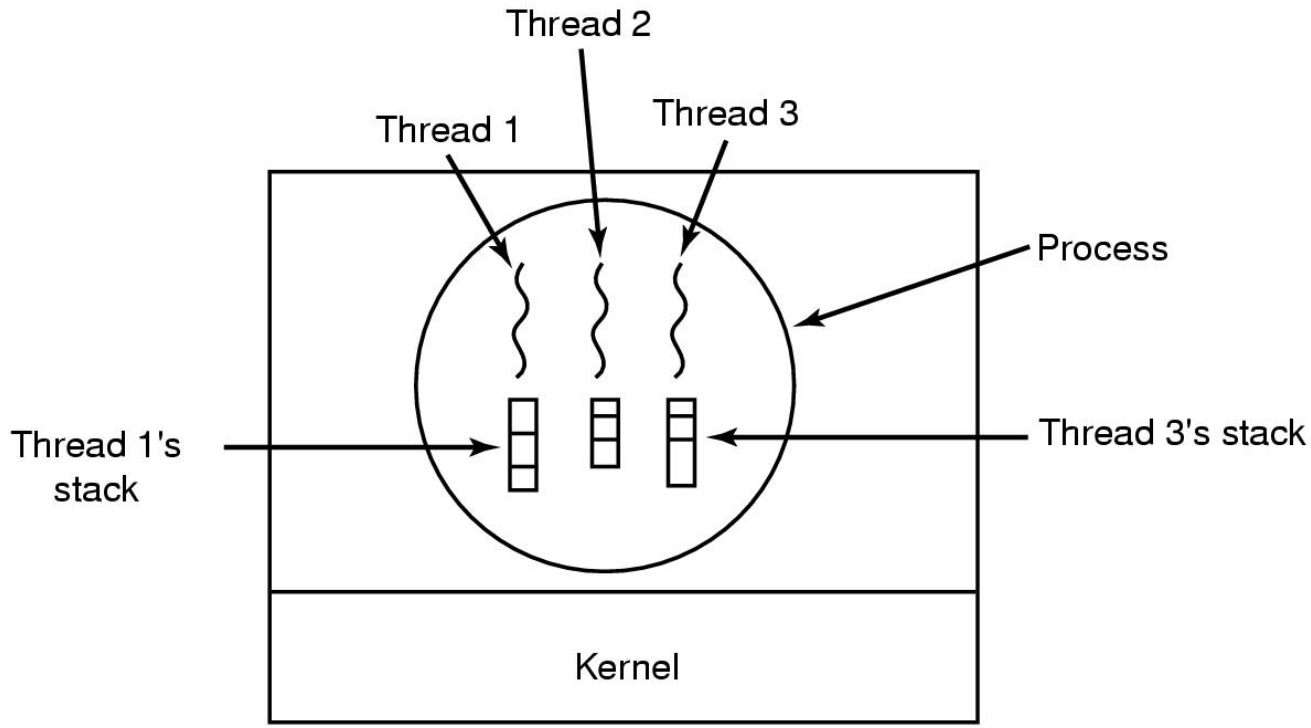
Threads

Várias threads podem compartilhar o mesmo espaço de endereçamento



Tanenbaum: Figura 2.6

Pilhas independentes



Tanenbaum: Figura 2.8

Como trabalhar com threads

Veja os comandos:

- `pthread_create`
- `pthread_join`
- `pthread_exit`

Para mais informações: `man <comando>`

Como criar uma thread

```
int  pthread_create(pthread_t  *thread,
                    pthread_attr_t *attr,
                    void * (*start_routine)(void *),
                    void *arg);
```

Veja o código: `create0.c` ou
vamos fazer uma revisão sobre apontadores para funções em C?

Apontadores para funções em C

- Veja o código `fs.h`

```
void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

- `base`: apontador para a base do vetor
- `nmemb`: número de elementos
- `size`: tamanho (em bytes) de cada elemento
- `compar`: apontador para uma função que compara elementos e pode retornar 0, valores positivos ou negativos conforme o resultado da comparação.

- Veja o código: `qsort.c`

Como esperar por uma thread

```
int pthread_join(pthread_t thr,
                 void **thread_return);
```

Veja os códigos: join0.c

Como passar argumentos para uma thread

- Exemplo: cada thread pode precisar de um identificador único.
- Veja os códigos: create1.c, create2.c, create3.c, create4.c e create5.c

Como encerrar a execução de uma thread

- Comando `return` na função principal da thread (passada como parâmetro em `pthread_create`)
- Análogo ao comando `return` na função `main()`

Veja os códigos: return0.c, return1.c pthread_return.c

Como encerrar a execução de uma thread

- `void pthread_exit(void *retval);`
- Análogo ao comando `exit(status);`

Veja os códigos: exit0.c, exit1.c e pthread_exit0.c

Vamos examinar a pilha de execução

- Componentes do frame (não necessariamente nesta ordem):
  - valor de retorno
  - endereço de retorno
  - registradores
  - argumentos para a função
  - variáveis locais da função
- No gdb execute comandos do tipo:

```
(gdb) p \x (int[10]) *s1
(gdb) p \x (int[10]) *(s1-2)

(gdb) set *(s1-2) = ...
```

Pilhas de execução

- Veja as pilhas de execução: pilhas.c
- Você acha que uma thread pode . . .
  - ler o conteúdo da pilha de outra thread?
  - escrever na pilha de outra thread?
  - matar a outra thread? veja o código corrompe-pilhas.c

Treinando...

Você já pode escrever código multithread!

- Escreva códigos para soma, subtração e multiplicação de matrizes.
- Veja o laboratório proposto para a turma de 2010: *Pesca-palavras*