

MC504/MC514 - Sistemas Operacionais

Processos e Threads: Introdução

Islene Calciolari Garcia

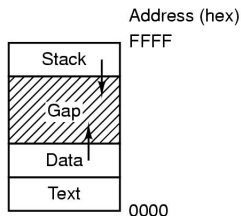
Segundo Semestre de 2016

Sumário

- 1 Espaço de endereçamento
- 2 Pthreads
 - create
 - Apontadores genéricos e apontadores para função em C
 - join
 - Argumentos
 - exit
 - Exemplo
- 3 Pilhas de execução
- 4 Dicas GDB

Processo

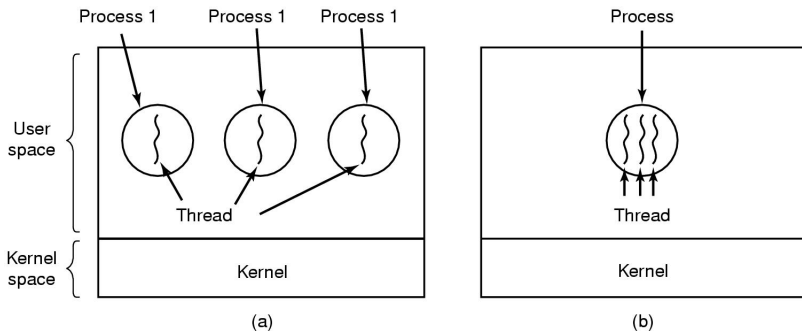
- Programa em execução
- Espaço de endereçamento



Tanenbaum: Figura 1.20

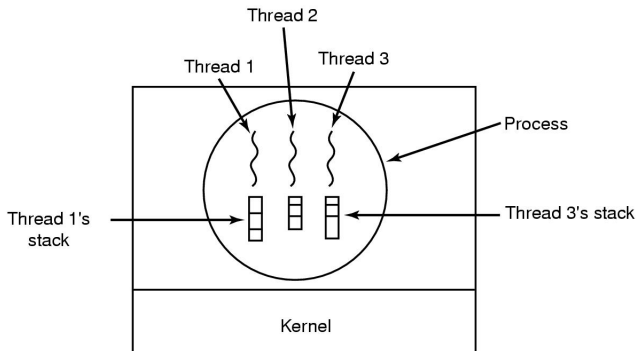
Threads

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` e `executa-funcao.c`

join

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

exit

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`

exit

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`

Create e Join

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

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

Veja o código: `create_join.c`

Pilhas de execução

- Veja as pilhas de execução: `endereco-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`

GDB e threads

- `info threads`
- `thread <thread number>`
- `set non-stop on`

In non-stop mode, when one thread stops, other threads can continue to run freely. You'll be able to step each thread independently, leave it stopped or free to run as needed.

- `interrupt`

Interrupt the execution of the debugged program. If non-stop mode is enabled, `interrupt` only the current thread, otherwise all the threads in the program are stopped. To interrupt all running threads in non-stop mode, use the `-a` option.