

MC514–Sistemas Operacionais: Teoria e Prática
1s2009

Processos e Threads 2

Objetivos

- Pthreads
 - Revisão create e join
 - Operação exit
- Pilha de execução

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`

Como passar argumentos para uma thread

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

Pilha de execução:

- Espaço para valor de retorno da função
- Argumentos
- Endereço de retorno
- Registradores
- Variáveis locais

Veja o código: `pilha.c`

Exemplo de endereços

	⋮
0x804971c	global
	⋮
0x80483ed	main
0x80483a4	f

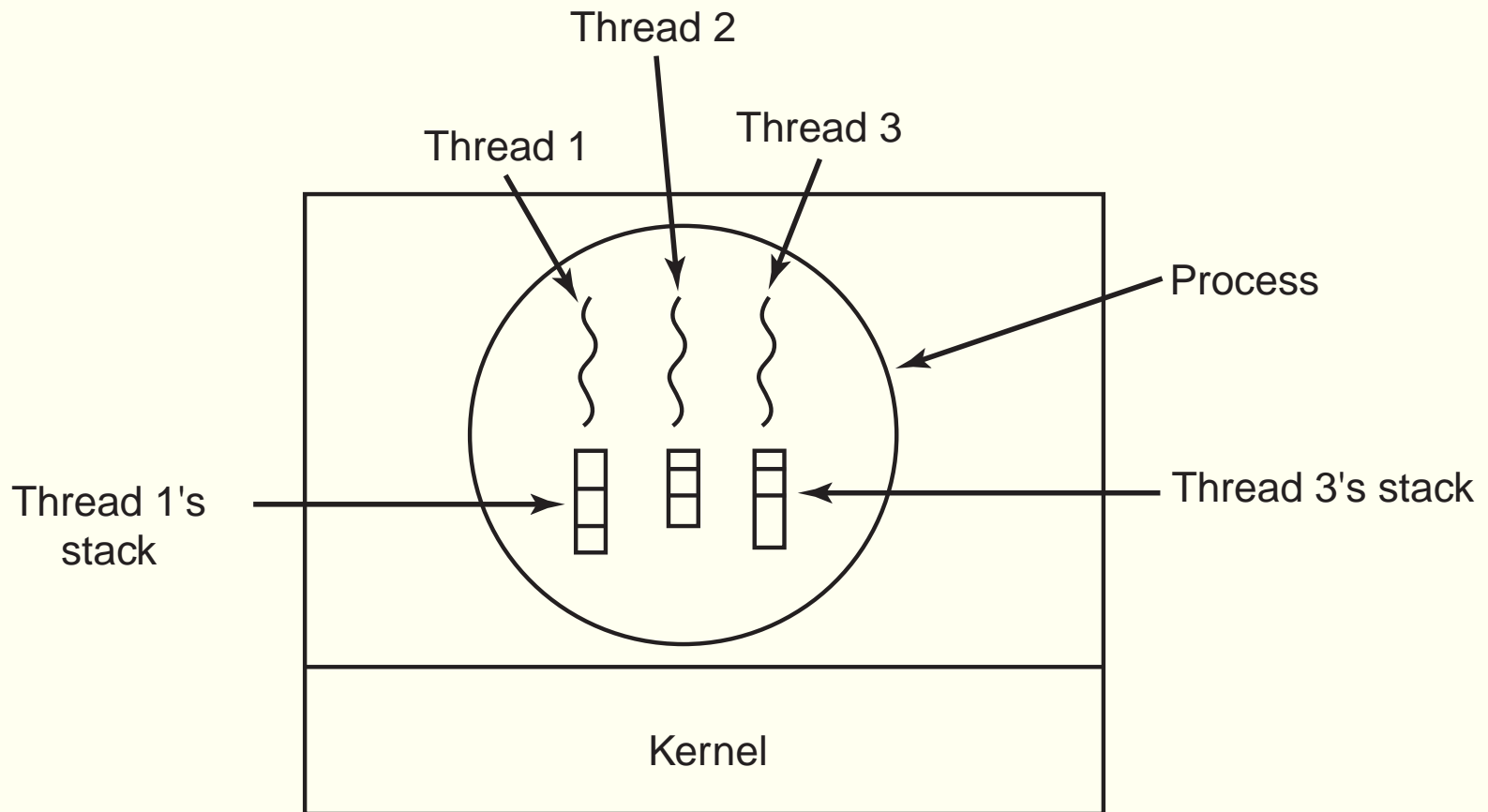
Exemplo de endereços

0xbfb0f1f0	local_main
	⋮
0xbfb0f1d0	param_f
	⋮
0xbfb0f1c4	v[1]
0xbfb0f1c0	v[0]
	⋮

É muito fácil corromper a pilha

- Basta fazer acesso a posições não alocadas de um vetor
- Veja os códigos: `corrompe_pilha.c` e `corrompe_pilha1.c`

Pilhas independentes



Tanenbaum: Figura 2.8

Veja o código: pilhas.c

Uma thread pode corromper a pilha de outra thread

- Pilhas são independentes, mas não protegidas
- Veja o código: `corrompe_thread.c`