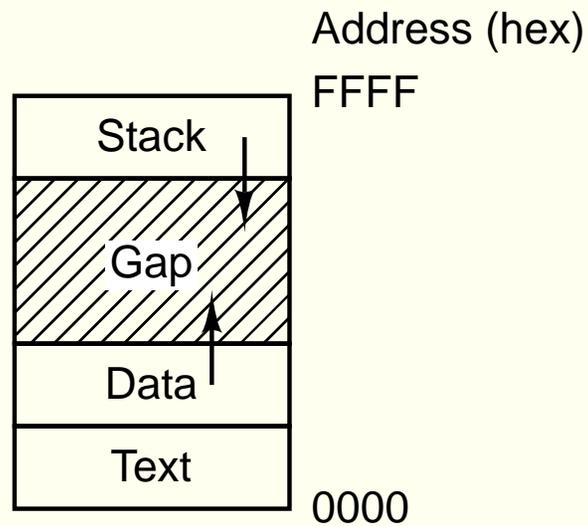


Processos e Threads

Aula 2

Processo

- Programa em execução
- Espaço de endereçamento



Veja o código: `ender2.c`

Exemplo de endereços

	⋮
0x804968C	global
	⋮
0x80483C4	main
0x8048378	f

Exemplo de endereços

0xBFE7CB94	local_main
	⋮
0xBFE7CB70	param_f
	⋮
0xBFE7CB64	v[1]
0xBFE7CB60	v[0]
0xBFFFF680	⋮

O que é armazenado na pilha?

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

É muito fácil corromper a pilha

- Basta fazer acesso a posições não alocadas de um vetor;
- Veja o código: `corrompe_pilha.c` e `loop.c`

Uma thread pode corromper a pilha de outra thread

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

Gerência de recursos para processos e threads

Compartilhados	Independentes
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	

Pthread_join

- Uma thread pode esperar pelo término de uma thread irmã.
- Veja o código `deadlock.c`

Escalonamento de threads

- A execução de uma thread pode ser interrompida a qualquer momento.
- Veja o código `preemptivo.c`

Threads podem dividir trabalho

```
#define MAX_VEZES 5

void* f_thr(void *v) {

    while(n_vezes) {    /* BUG */
        sleep(1);
        faz_alguma_coisa();
        n_vezes--;
    }
    return NULL;
}
```

- Veja o código `divisao_com_bug.c`

Condição de disputa

Saída esperada

```
int s = 0; /* Variável compartilhada */
```

Thread 0

- (i) `s = 0;`
- (ii) `print ("Thr 0: ", s);`

Thread 1

- (iii) `s = 1;`
- (iv) `print ("Thr 1: ", s);`

Saída: Thr 0: 0
Thr 1: 1

Condição de disputa

Saída esperada II

```
int s = 0; /* Variável compartilhada */
```

Thread 0

(iii) `s = 0;`

(iv) `print ("Thr 0: ", s);`

Thread 1

(i) `s = 1;`

(ii) `print ("Thr 1: ", s);`

Saída: Thr 1: 1
Thr 0: 0

Condição de disputa

Saída inesperada

```
int s = 0; /* Variável compartilhada */
```

Thread 0

- (i) `s = 0;`
- (iii) `print ("Thr 0: ", s);`

Thread 1

- (ii) `s = 1;`
- (iv) `print ("Thr 1: ", s);`

Saída: Thr 0: 1
Thr 1: 1

Veja o código: `inesperada.c`