Gerência de Memória

Sistemas Operacionais
2017-1
Flavio Figueiredo (http://flaviovdf.github.io)

Até Agora

- Aprendemos que o SO é uma interface entre o hardware eo software
  - Trocas de contexto
  - Gerência de Interrupções
  - Chamadas de sistemas

- *Processo:* Abstração base de sistemas operacionais
  - Escalonamento: Qual processo vai executar e quando
  - Threads: Paralelismo e compartilhamento de dados dentro de um processo
  - Sincronização: Como fazer a comunicação entre processos

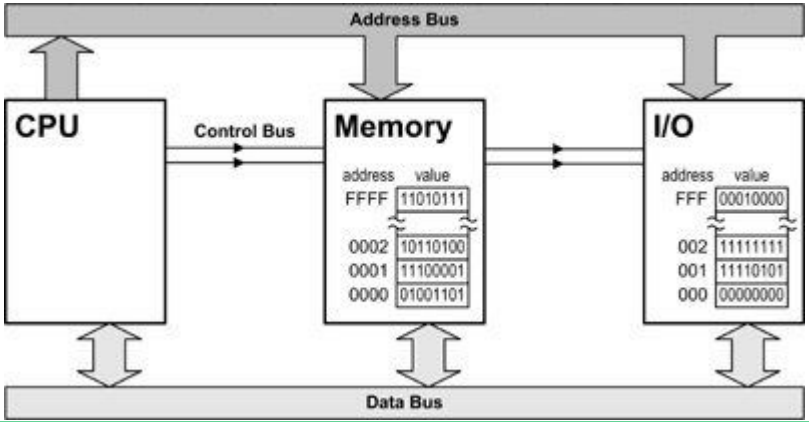
Novo problema...

Como referenciar a memória?

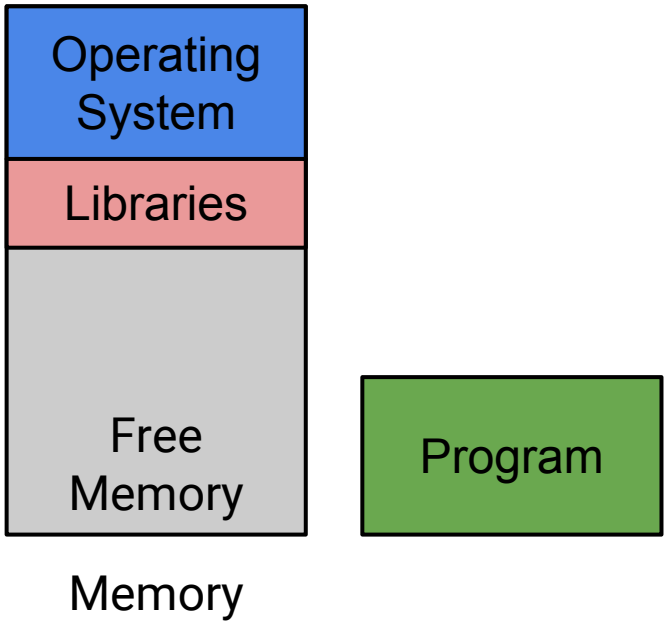
Novo problema...

Como referenciar a memória?

```
load
store
jmp/move
```

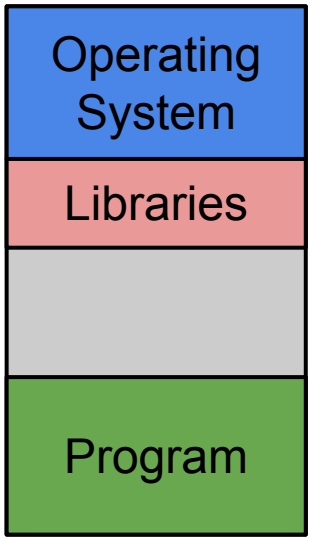


Se Removermos Multiprogramação



- Podemos utilizar endereços absolutos
- Programa pode ocupar todo espaço

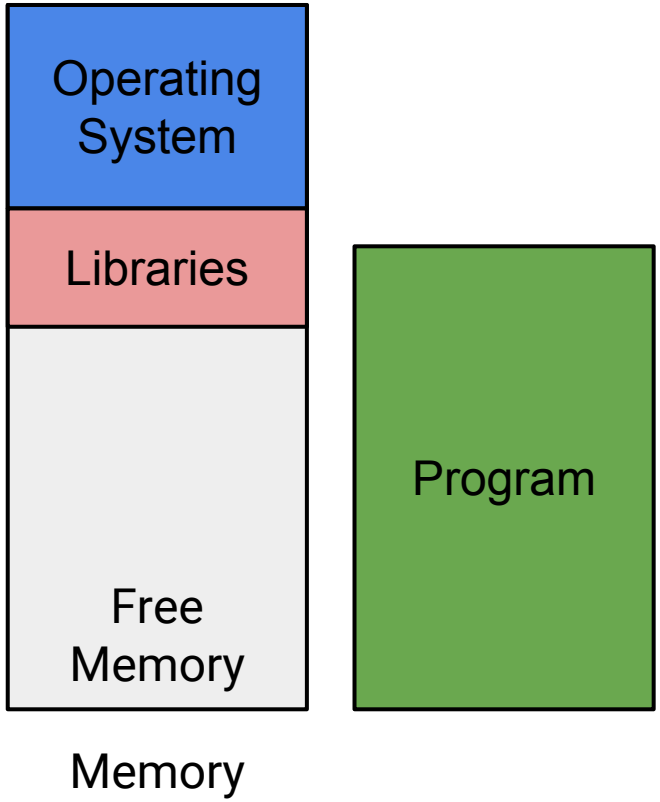
Se Removermos Multiprogramação



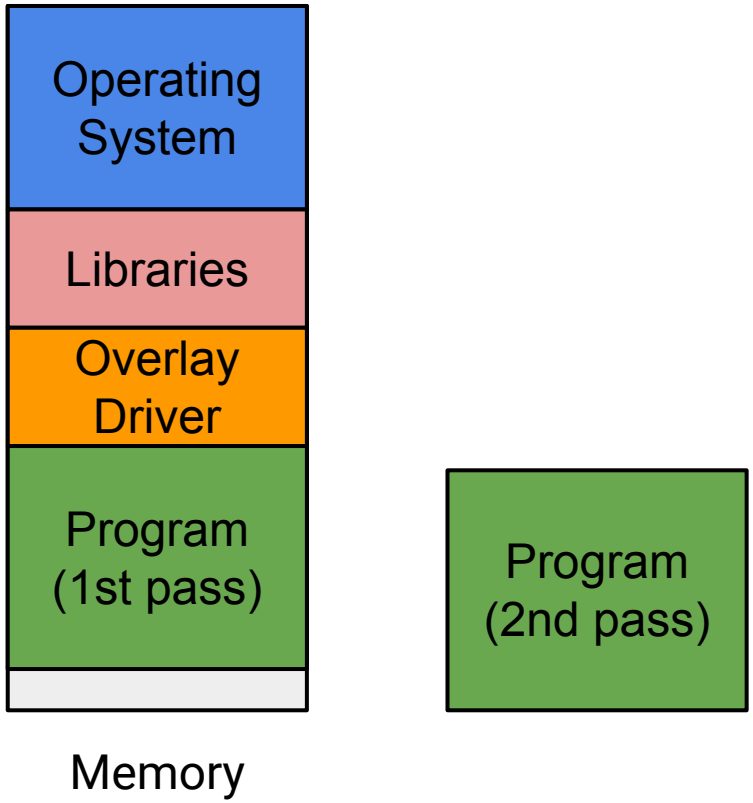
- Podemos utilizar endereços absolutos
- Programa pode ocupar todo espaço
- Endereços podem iniciar em 0x0000
- Problemas:
  - Garantir que processos não escrevam na área do SO/Libraries
  - Perdemos tudo que aprendemos em Processos

E se o programa não “encaixar”?

E se o programa não “encaixar”?

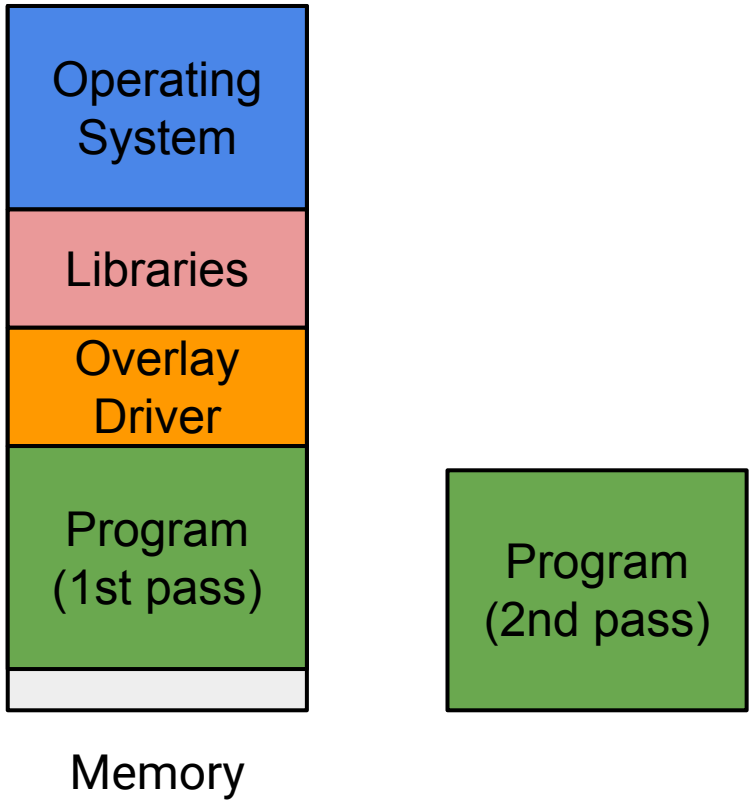


E se o programa não “encaixar”?



- Overlay driver para carregar partes do programa
  - Compilador:
    - 1st: Análise Sintática
    - 2nd: Análise Semântica

E se o programa não “encaixar”?

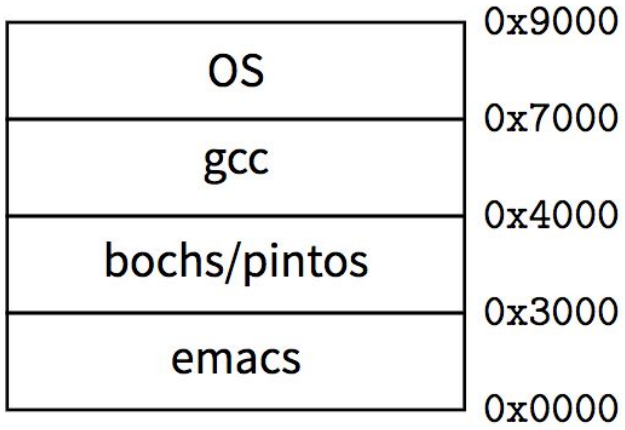


- Overlay driver para carregar partes do programa
- E a outra parte fica aonde?
  - Ainda mais se precisarmos chavear entre elas
  - MergeSort
    - “Divide” e “Conquer” são chamados de forma intercaladas

+ 1 problema...

Como fazer com que processos coexistam na memória?

Multiprogramação na memória

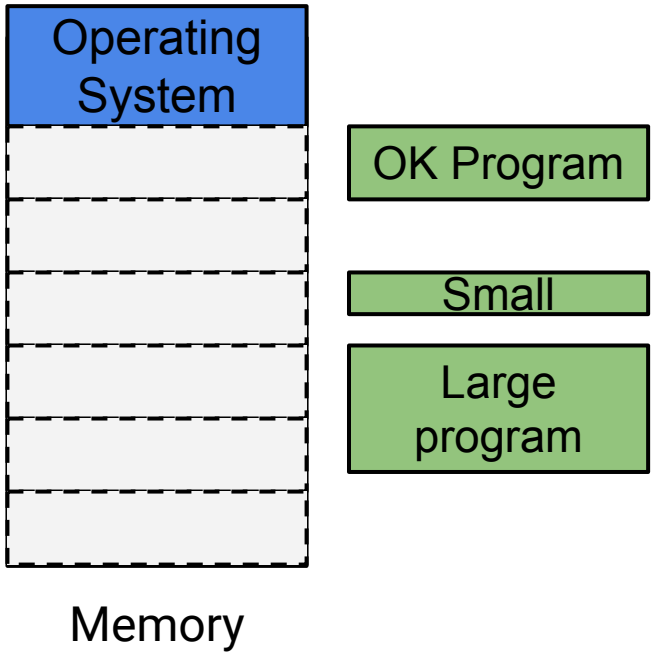


E se...

- GCC precisar de mais memória?
- Emacs escrever em 0x3001
- Alguém abrir o Chrome?

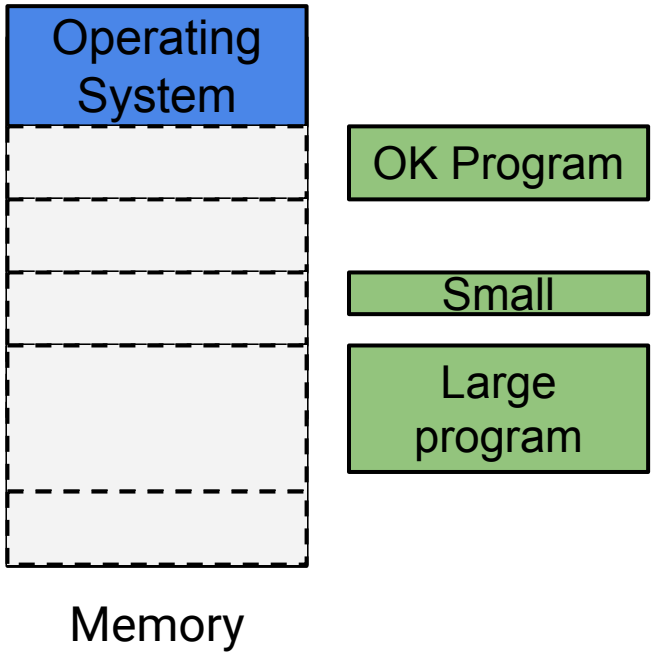
Segunda Solução

- Blocos de tamanho fixo de memória
- Problemas?



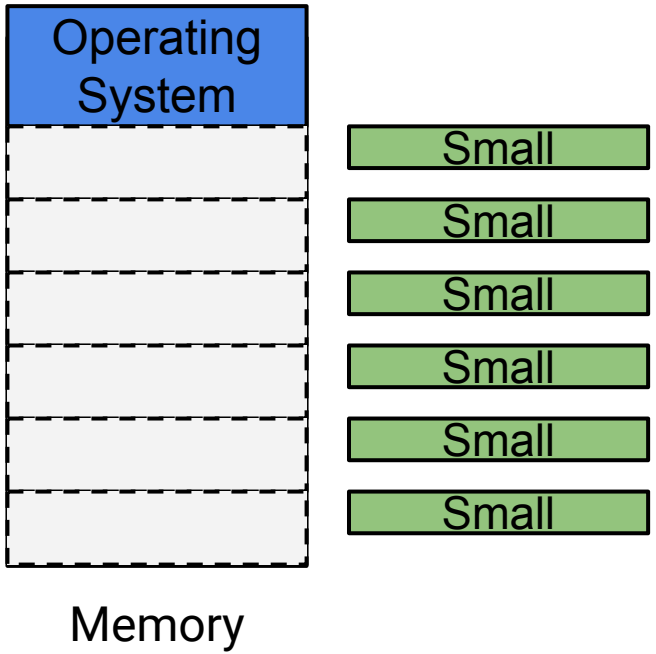
Segunda Solução

- Blocos de **múltiplos de** tamanho fixo de memória
- Problemas?



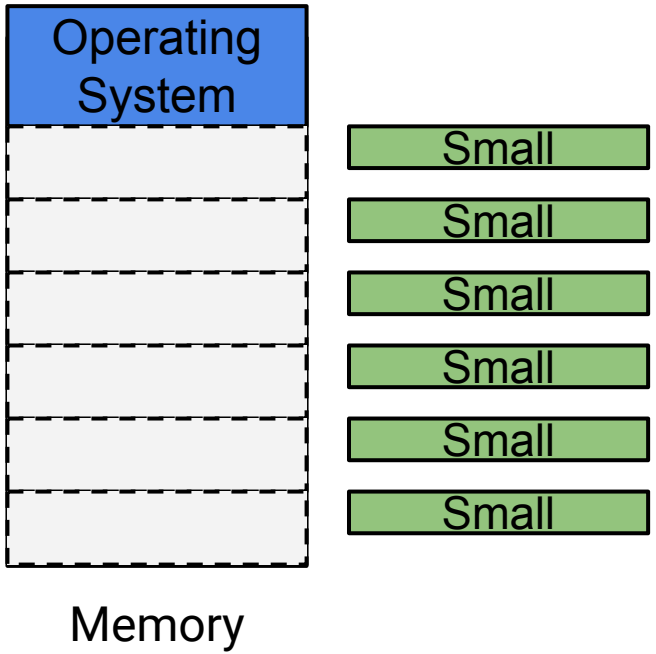
Segunda Solução

- Blocos de **múltiplos de** tamanho fixo de memória
- Problemas?



Segunda Solução

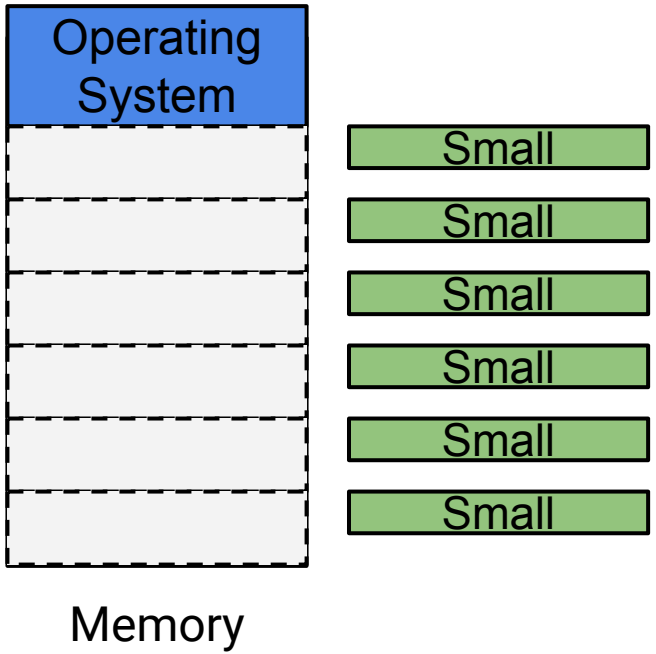
- Blocos de **múltiplos de** tamanho fixo de memória
- Problemas?



Como os processos endereçam?

Segunda Solução

- Blocos de **múltiplos de** tamanho fixo de memória
- Problemas?



Como os processos endereçam?

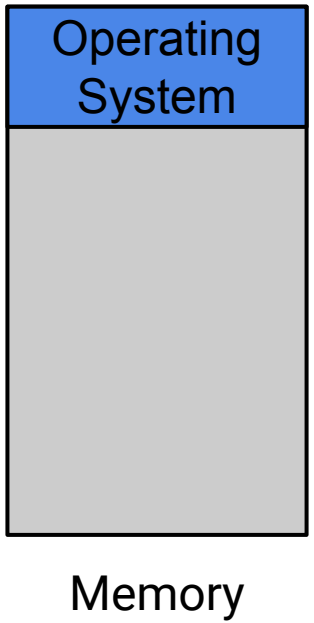
Cada um vai saber o bloco onde executa?

+ Problemas

- Quem deve fazer a tradução de endereços?

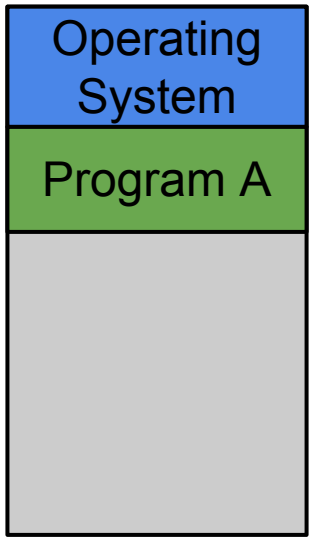
Terceira Solução: Alocação Contígua

- Não precisamos dividir a memória
- Como executar múltiplos programas?



Terceira Solução

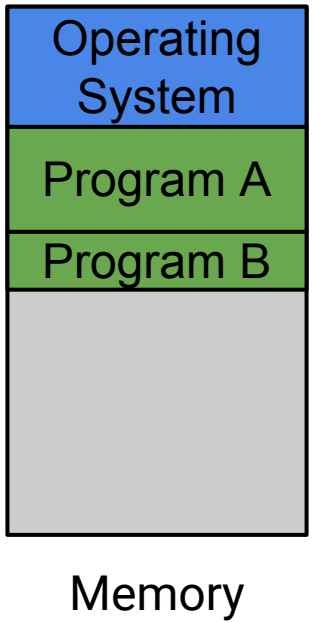
- Não precisamos dividir a memória
- Como executar múltiplos programas?



Memory

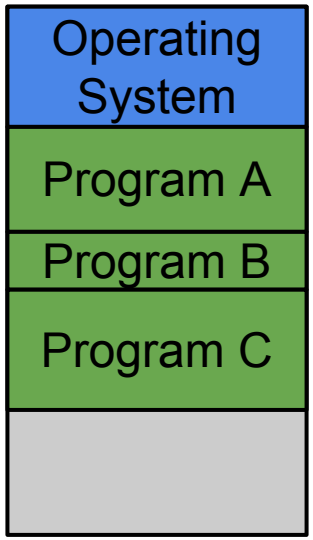
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



Terceira Solução

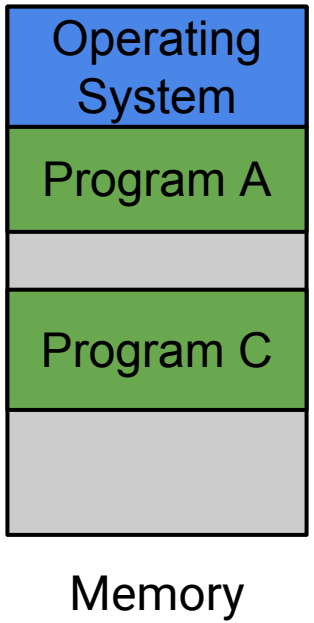
- Não precisamos dividir a memória
- Como executar múltiplos programas?



Memory

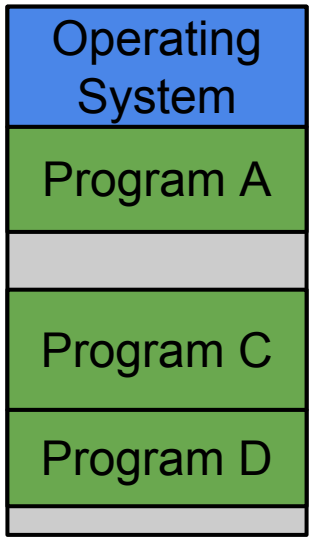
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



Terceira Solução

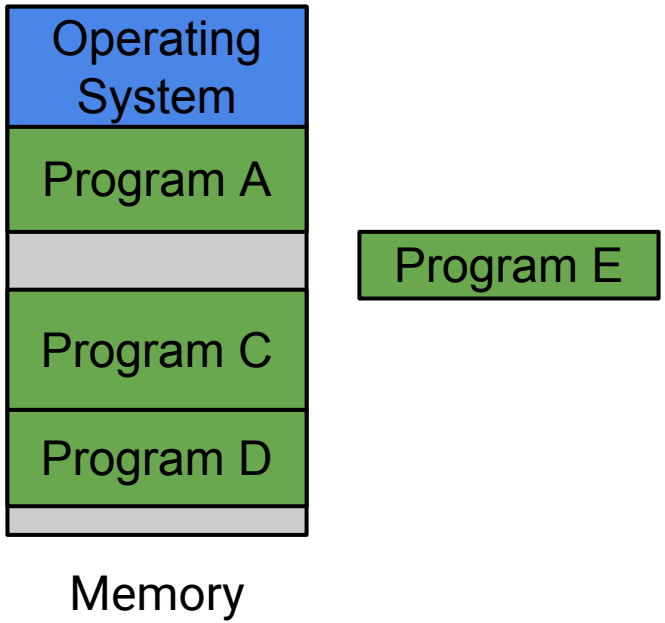
- Não precisamos dividir a memória
- Como executar múltiplos programas?



Memory

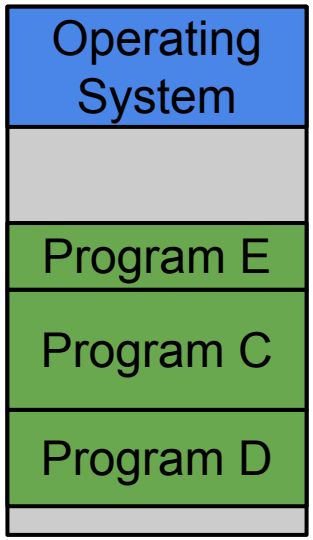
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



Terceira Solução

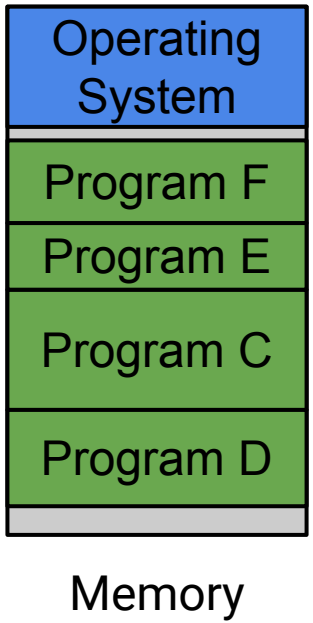
- Não precisamos dividir a memória
- Como executar múltiplos programas?



Memory

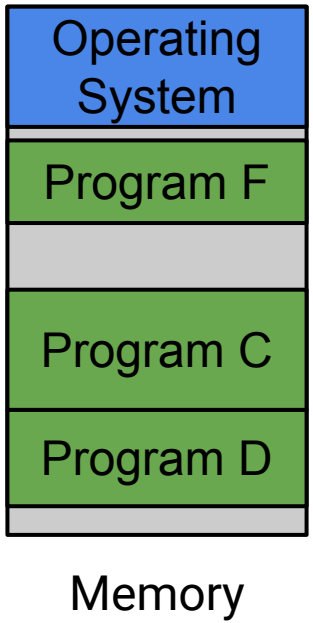
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



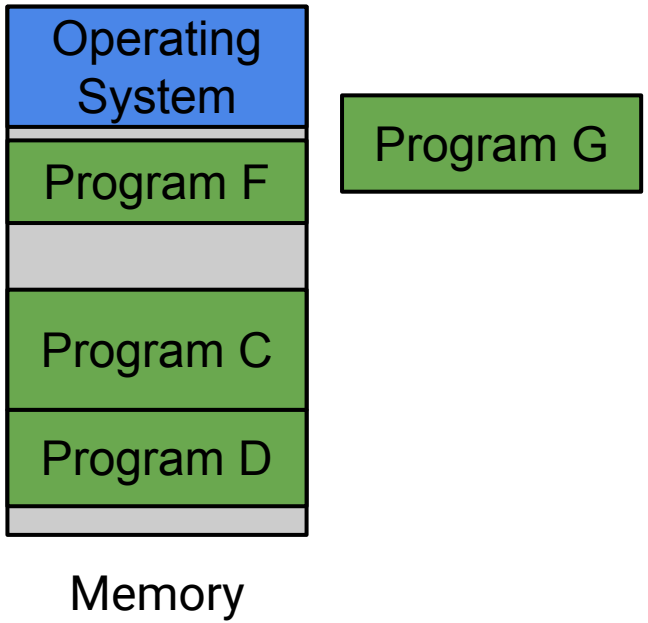
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



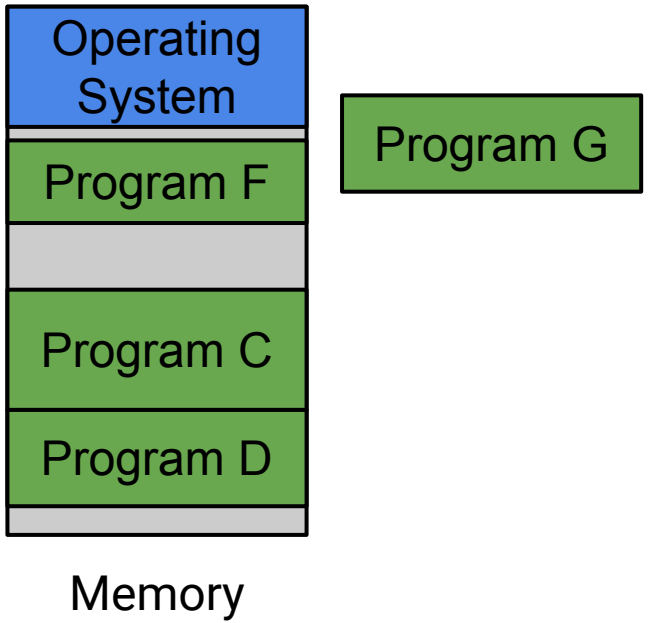
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



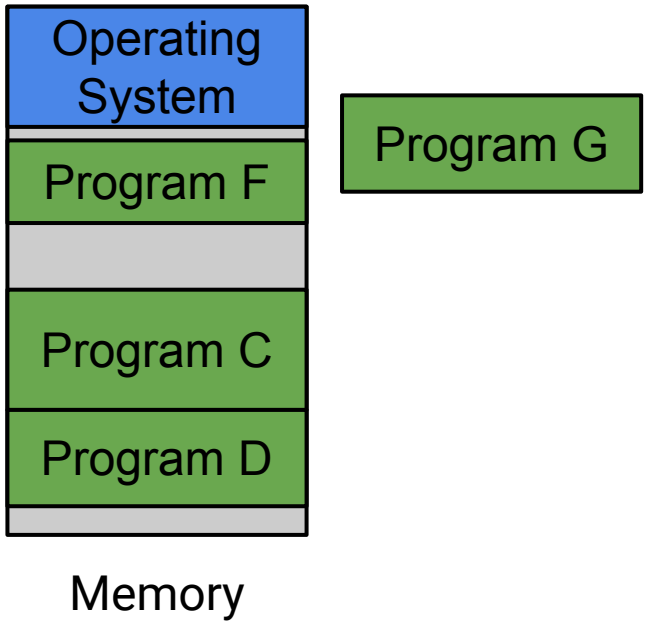
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



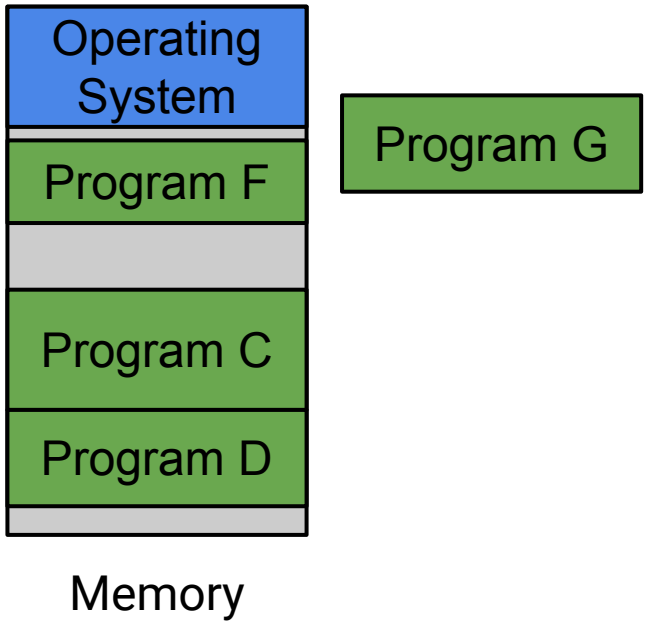
Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



Terceira Solução

- Não precisamos dividir a memória
- Como executar múltiplos programas?



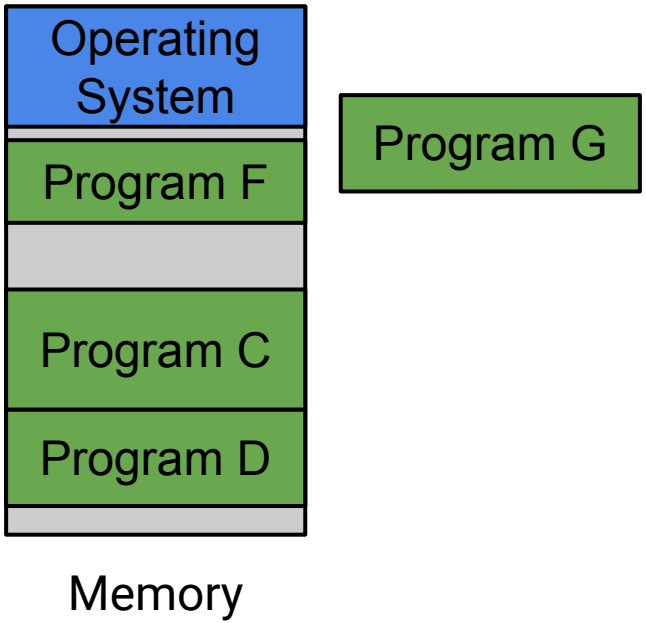
Fragmentação

- Com o tempo vamos ficando com buracos na memória
- Como fazer melhor uso de tal espaço?

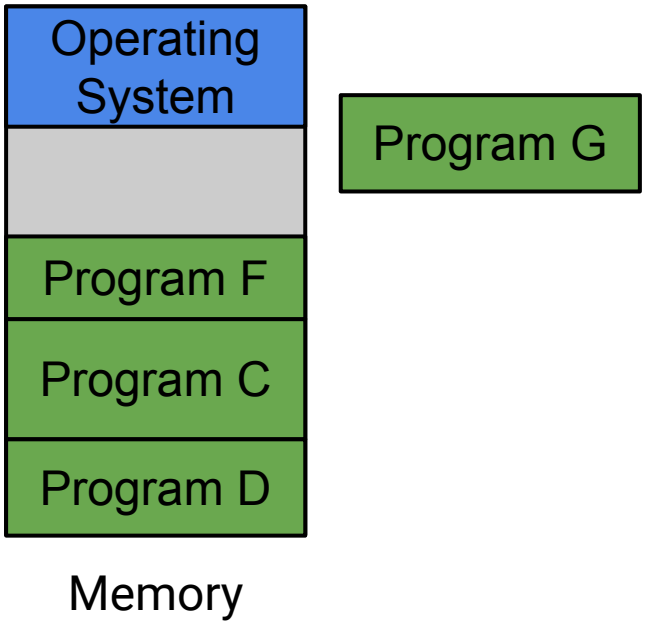
Fragmentação

- Com o tempo vamos ficando com buracos na memória
- Como fazer melhor uso de tal espaço?
- Compactação
  - Re-alocar os processos

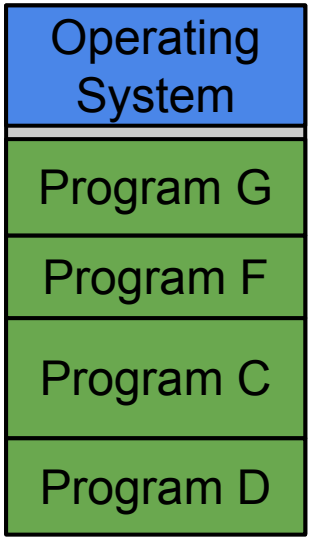
Compactando



Compactando



Compactando



Memory

+ Problemas: Fragmentação Interna

- Ainda assim o programa pode estar desperdiçando espaço internamente

- Heap
  - Precisa de espaço para crescer que pode estar sendo mal utilizado

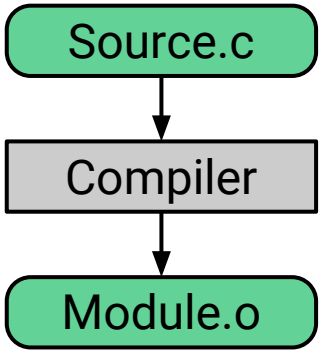
- Pilha
  - Precisa de espaço para crescer que pode estar sendo mal utilizado

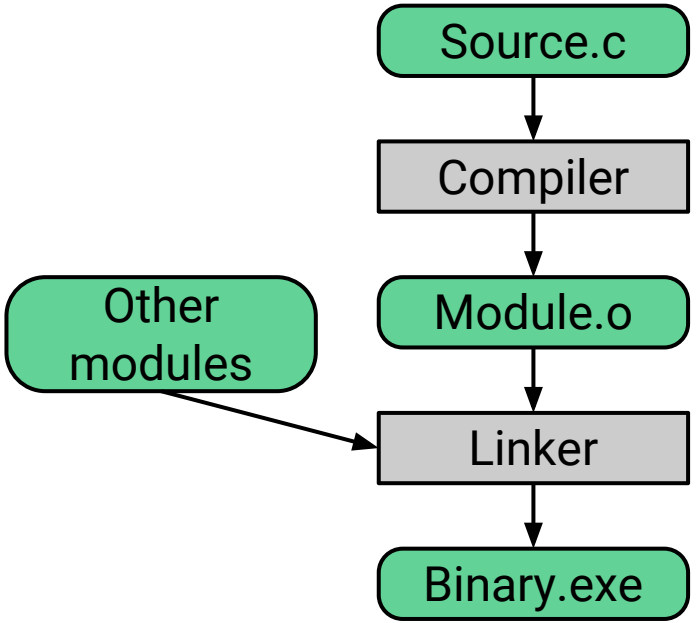
Traduzindo Endereços

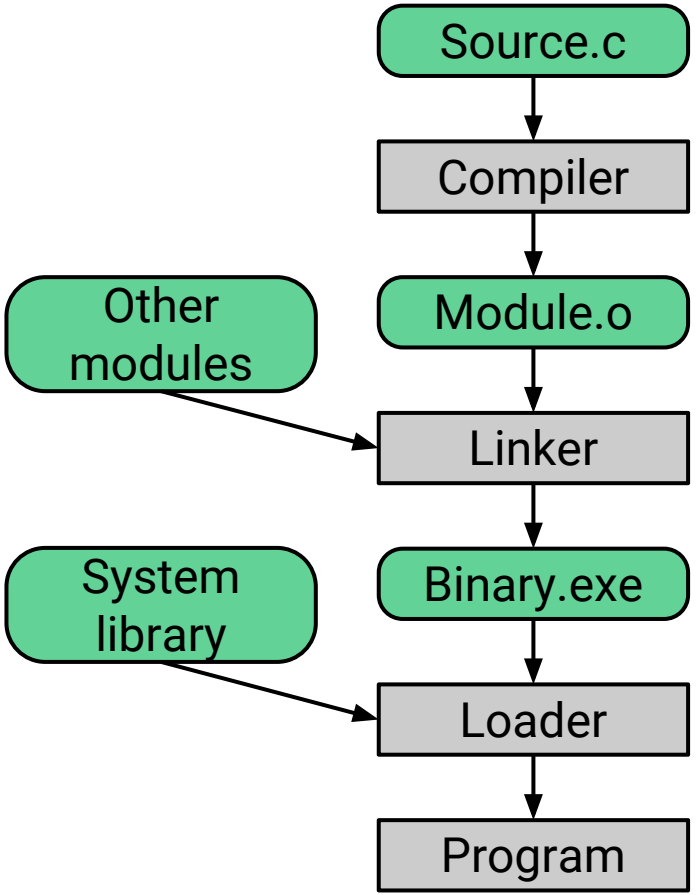
- Ainda não respondemos bem a seguinte pergunta
  - Quem deve fazer a tradução de endereços?

Compilando um Programa

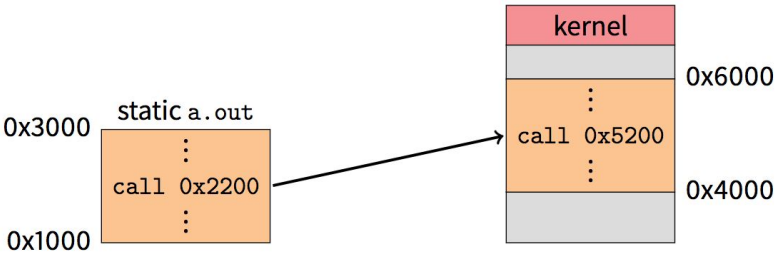
Source.c

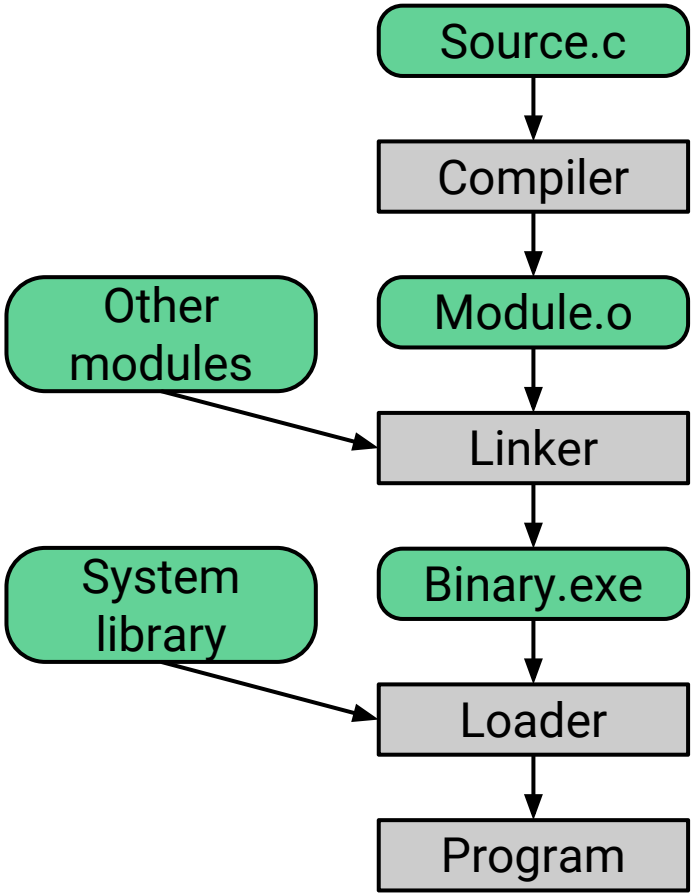

Source.c
Compiler
Module.o


Source.c
Compiler
Other modules
Module.o
Linker
Binary.exe

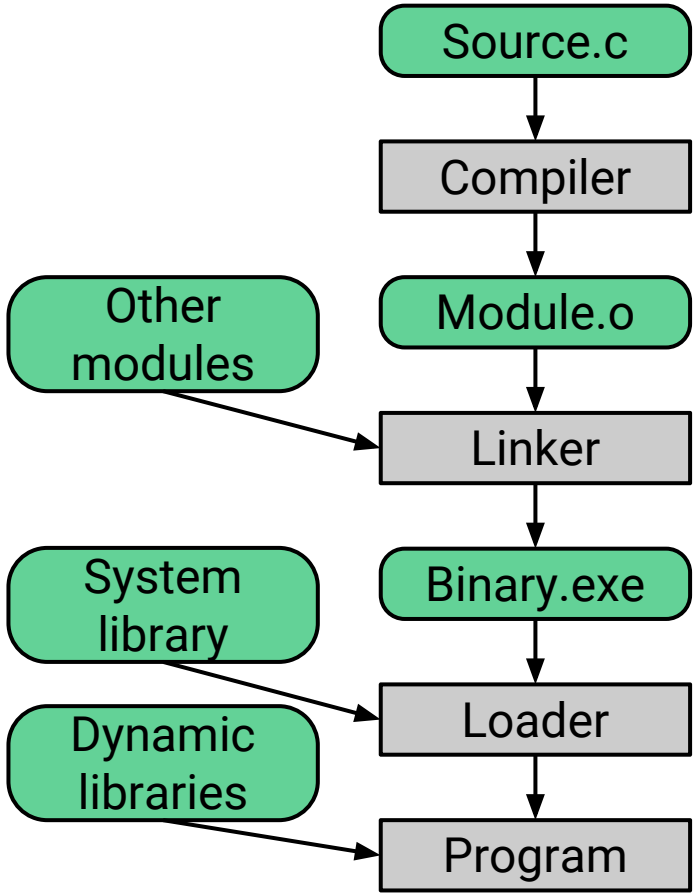

Source.c
Compiler
Other modules
Module.o
Linker
System library
Binary.exe
Loader
Program

- Uma ideia:
  - O loader pode traduzir endereços
  - Ao executar o programa re-escrever instruções

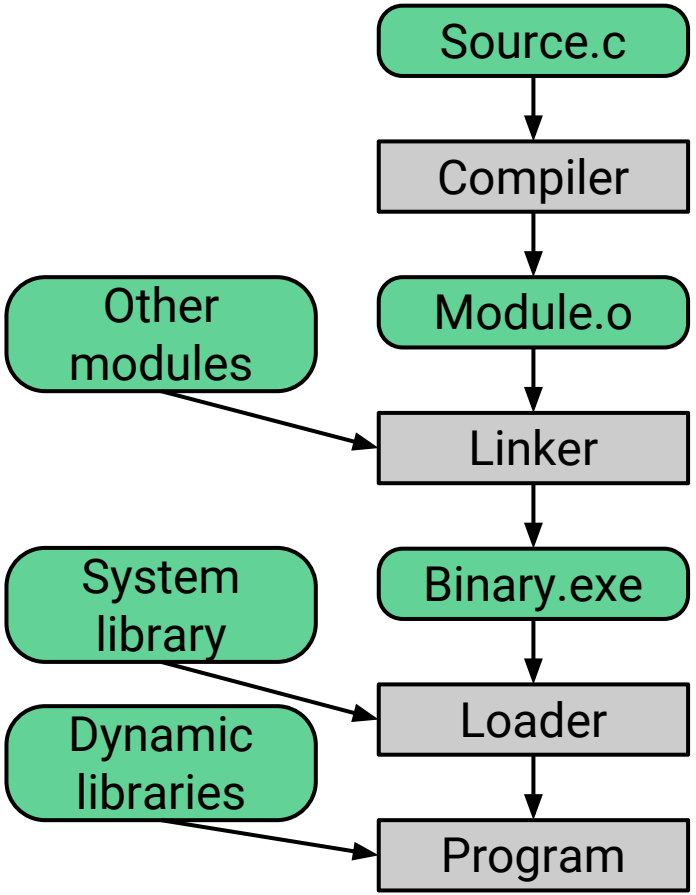




- Uma ideia:
  - O loader pode traduzir endereços
  - Ao executar o programa re-escrever instruções
  - Bibliotecas dinâmicas
    - Como traduzir?



- Uma ideia:
  - O loader pode traduzir endereços
  - Ao executar o programa re-escrever instruções
  - Bibliotecas dinâmicas
    - Como traduzir?
  - Re-escrever operações do programa
    - Custoso
  - Nada impede que um programa acesse regiões ilegais



Requisitos

- **Proteção**
  - Um processo não deve escrever no espaço de outro
- **Transparência**
  - Processos não devem trabalhar com endereços absolutos
  - De qualquer forma, o processo tem que ter a ideia de memória contínua
- **Recursos suficientes**
  - O programador e o processo podem assumir que memória é infinita
  - O SO que se vire para tratar disto
  - Disco ajuda
  - No mundo ideal, o programador não se preocupa com memória

Exemplo

https://github.com/flaviovdf/SO-2017-1/tree/master/examples/virtaddr

+ Problemas

- Quem deve fazer a tradução de endereços?
- SO?
- Usuário?
- Hardware?

+ Problemas

- Quem deve fazer a tradução de endereços?

- SO
  - Da para ser feito, mas pode ser lento
  - Operações de soma em cada operação que acessa memória

- Usuário
  - Não é uma boa opção

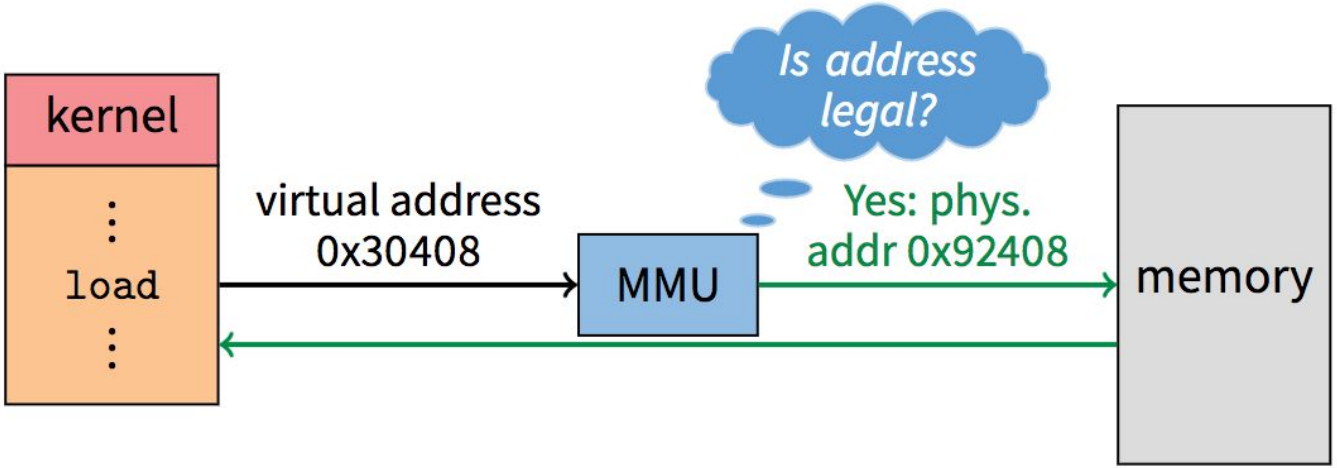
- **Hardware**
  - Melhor opção
  - [Primeira Solução] Registradores **base** e **limite**

Endereços Virtuais

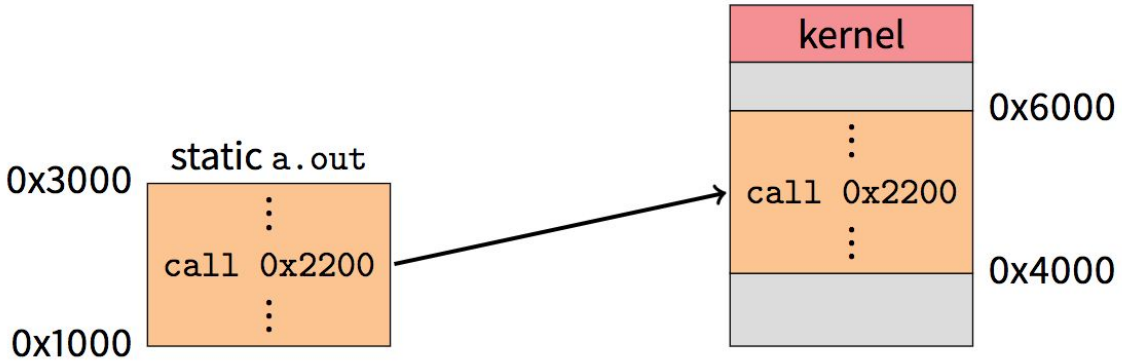
- Programas nunca referenciam a memória *per-se*
- Fazem uso de endereço virtual
- Traduzido para endereços reais
- Tradução deve ser eficiente
    - Hardware como falamos

Unidade Gerenciadora de Memória

Memory Management Unit (MMU)

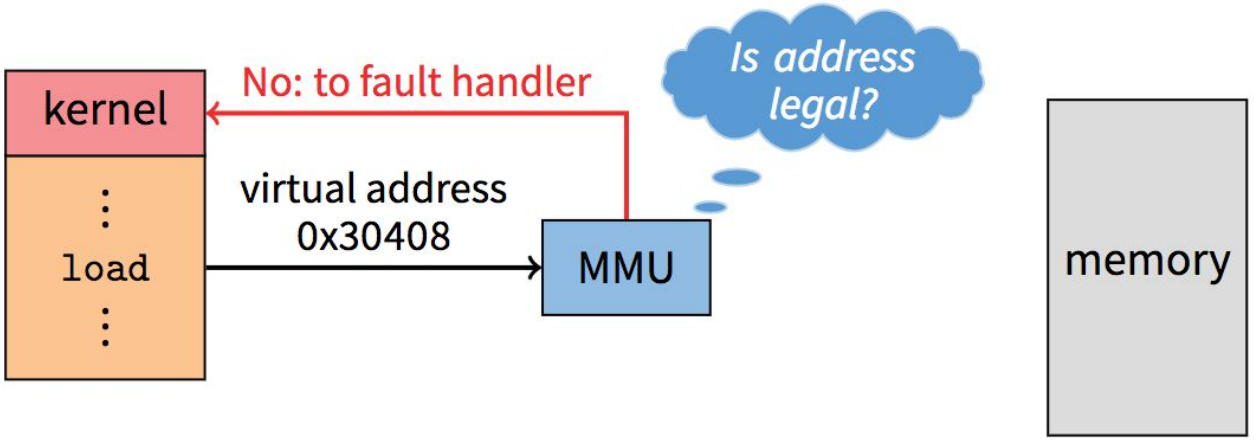


Registradores Base e Limite



- Troca de contexto mais simples
  - Atualizar 2 registradores
- Traduções em hardware
  - Endereço = Endereço Requisitado + **Base**
  - **Limite** cuida de acessos ilegais

Registradores Base e Limite



Novamente o Hardware Ajudando...

Hardware Requirements	Notes
Privileged mode	*Needed to prevent user-mode processes from executing privileged operations*
Base/bounds registers	*Need pair of registers per CPU to support address translation and bounds checks*
Ability to translate virtual addresses and check if within bounds	*Circuitry to do translations and check limits; in this case, quite simple*
Privileged instruction(s) to update base/bounds	*OS must be able to set these values before letting a user program run*
Privileged instruction(s) to register exception handlers	*OS must be able to tell hardware what code to run if exception occurs*
Ability to raise exceptions	*When processes try to access privileged instructions or out-of-bounds memory*

Limitações de Base e Limite?

Limitações de Base e Limite?

- Em um *fork* criamos um clone do processo
  - Precisamos copiar tudo?

Limitações de Base e Limite?

- Em um *fork* criamos um clone do processo
  - Precisamos copiar tudo?

- Como lidar com a fragmentação interna?
  - Ainda mantemos aquele espaço entre a pilha e a fila

Limitações de Base e Limite?

- Em um *fork* criamos um clone do processo
  - Precisamos copiar tudo?
- Como lidar com a fragmentação interna?
  - Ainda mantemos aquele espaço entre a pilha e a fila
- E se um processo precisar de mais memória?
  - Malloc que ultrapassa o limite de base e limit

Memória de um Programa

- Várias partes
  - Code
  - Hooks for libraries
  - Data
  - Global variables
  - Stack
  - Heap

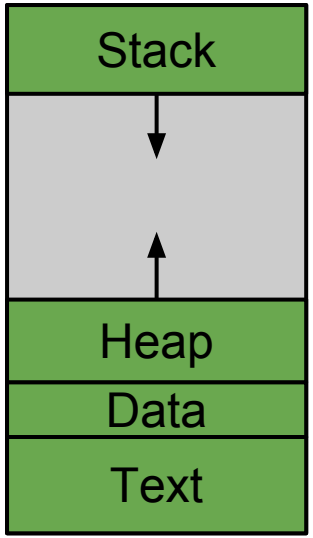
Memória de um Programa

- Programa
  - Code
  - Hooks for libraries
  - Data
  - Global variables
  - Stack
  - Heap

- Linux
  - Reference counters
  - List and tree of `vm_area`
  - Flags and basic segments
    - Code
    - Heap
    - Stack
    - Data
  - `struct vm_area`
    - Start/end
    - Protection
    - File pointer (mmap)

E aquela fragmentação interna?

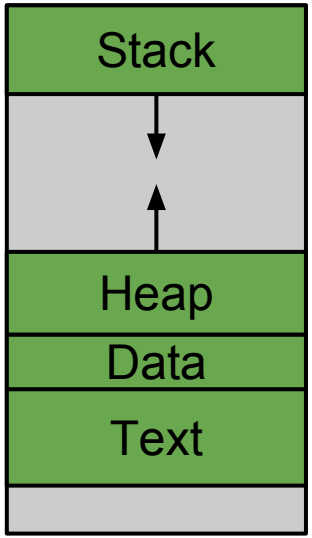
- Programa
  - Code
  - Hooks for libraries
  - Data
  - Global variables
  - Stack
  - Heap



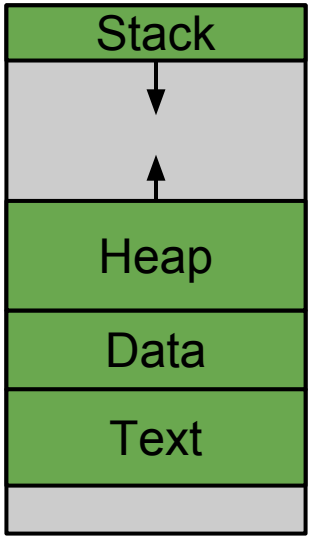
Program

- Linux
  - Reference counters
  - List and tree of `vm_area`
  - Flags and basic segments
    - Code
    - Heap
    - Stack
    - Data
  - `struct vm_area`
    - Start/end
    - Protection
    - File pointer (mmap)

Segmentação

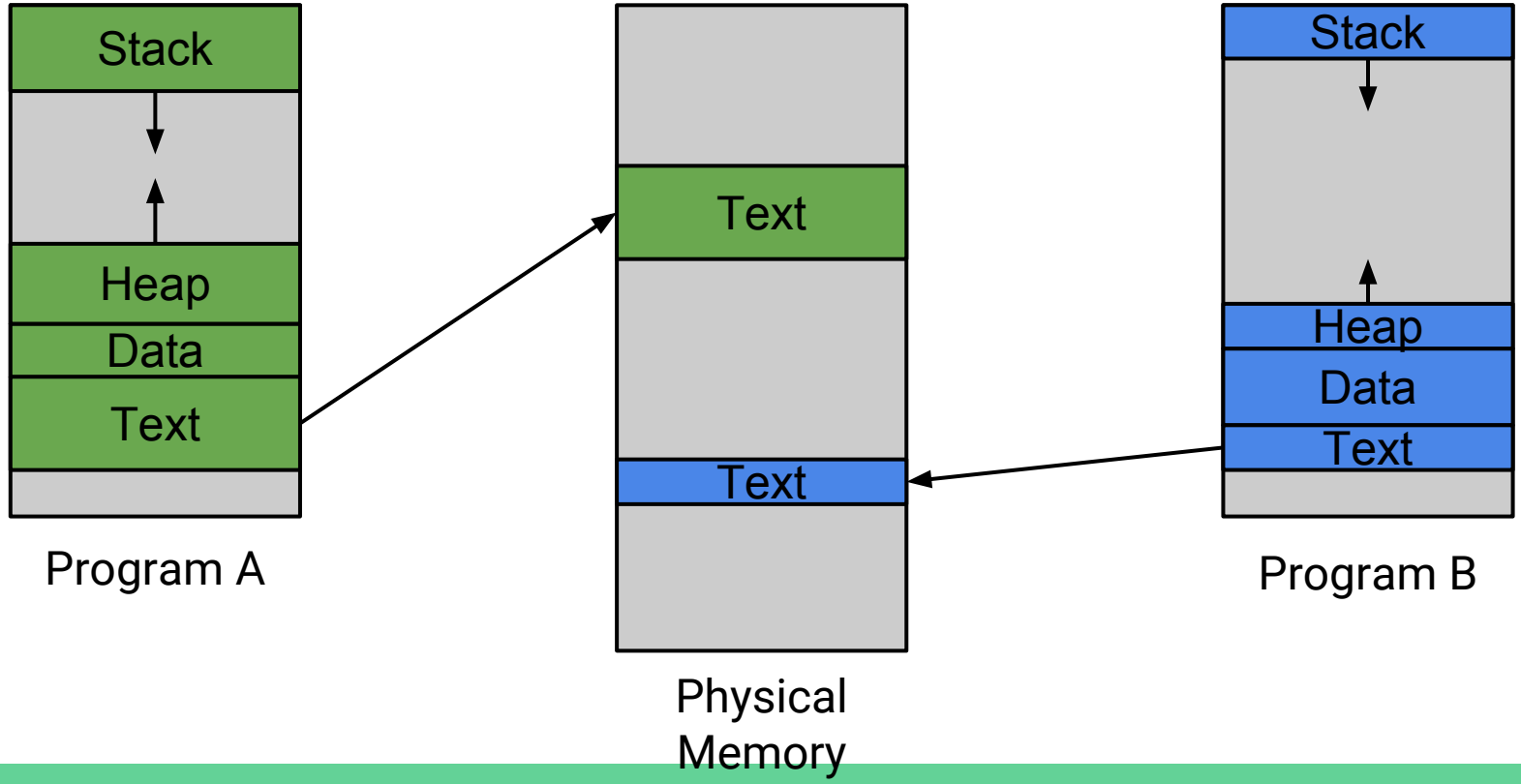


Program A

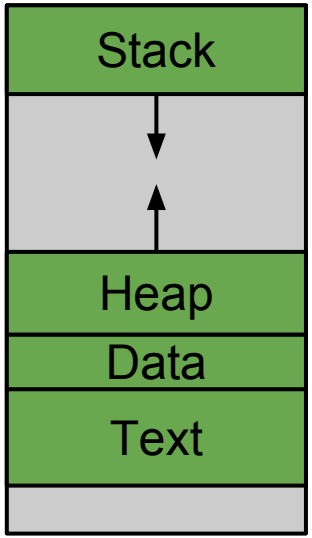


Program B

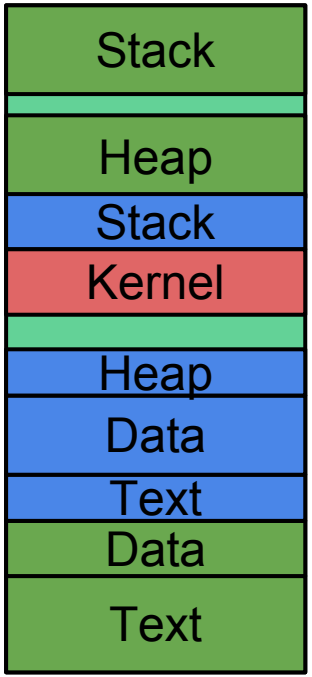
Segmentação



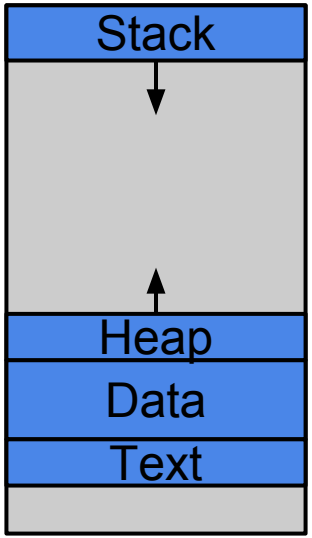
Segmentação



Program A



Physical Memory



Program B

Próximas aulas...

- Como implementar segmentação?
- Como lidar com fragmentação?
- Como fazer com que programas tenha a ideia de memória infinita?