Software Básico

Aula #02

Introdução à Execução de Programas

O que são arquivos fonte, objeto e executável? Como são organizados esses arquivos nos sistemas Linux?

Ciência da Computação - BCC2 - 2023/02
Prof. Vinícius Fülber Garcia

Arquivos de Programas

Quais são os tipos de arquivos com os quais você trabalha quando está **DESENVOLVENDO UM PROGRAMA**?

Arquivos de Programas

```c
#include <stdio.h>

int main(int argc, char **argv){
    printf("Hello World\n");
    return 0;
}
```

Vamos considerar a existência de um arquivo chamado “programa.c” contendo códigos para apenas exibir “Hello World” no terminal.

O que esse **arquivo contém**?
Como você **chama este arquivo**?

Arquivos de Programas

```c
#include <stdio.h>

int main(int argc, char **argv){
    printf("Hello World\n");
    return 0;
}
```

ARQUIVO FONTE

O arquivo fonte contém código legível para humanos relacionado a um programa.

Para isso, uma linguagem de programação é usada, tipicamente, explicitada na extensão do arquivo.

Arquivos de Programas

ARQUIVO FONTE

Sendo os arquivos fonte de um programa são organizados para compreensão humana, então visualizadores de texto são capazes de exibir os mesmos com sucesso!

```
$ cat programa.c
#include <stdio.h>

int main(int argc, char **argv){
    printf("Hello World\n");
    return 0;
}
```

Arquivos de Programas

ARQUIVO FONTE

Porém, a exibição por parte dos visualizadores/editores de texto fazem uma tradução dos dados considerando uma tabela que relaciona código com caracteres. Na verdade, os arquivos fonte armazenam **NÚMEROS**.

```
$ hexdump -C programa.c
00000000  23 69 6e 63 6c 75 64 65  20 3c 73 74 64 69 6f 2e  |#include <stdio.|
00000010  68 3e 0a 0a 69 6e 74 20  6d 61 69 6e 28 69 6e 74  |h>..int main(int|
00000020  20 61 72 67 63 2c 20 63  68 61 72 20 2a 2a 61 72  | argc, char **ar|
00000030  67 76 29 7b 0a 20 20 20  20 70 72 69 6e 74 66 28  |gv){.    printf(|
00000040  22 48 65 6c 6c 6f 20 57  6f 72 6c 64 5c 6e 22 29  |"Hello World\n")|
00000050  3b 0a 20 20 20 20 72 65  74 75 72 6e 20 30 3b 0a  |;.    return 0;.|
00000060  7d 0a                                            |}.|
00000062
```

Relembrando: Codificação de Dados

Conteúdo: Programação #2

- *American Standard Code for Information Interchange* (ASCII)
  - Um caractere conta com 7 bits (128 símbolos)
  - Extensões (ISO/IEC 8859)
- *Universal Character Encoding Standard* (Unicode)
  - Um caractere conta com 32 bits (4G símbolos, 149186 definidos)
  - Inicia com a tabela ASCII (primeiros 128 símbolos)
- *Universal Coded Character Set* (UTF8, UTF16, UTF32)
  - Utilização progressiva de bytes

Arquivos de Programas

No processo de compilação (particularmente no GCC), podemos optar pela execução completa de geração do executável:

```
gcc programa.c -o programa
```

Ou ainda, podemos gerar arquivos de código-objeto do programa em questão (muito útil quando existem múltiplos arquivos fonte compondo o programa):

```
gcc -c programa.c
```

Mas o que é e para que serve um código-objeto?

Arquivos de Programas

Um arquivo de código-objeto contém a "transcrição" de um fonte específico para código de máquina (após pré-processamento).

Porém, o programa não pode ser executado, uma vez que **o ligador ainda não atuou** para prover funções apenas referenciadas no arquivo fonte.

Mas e qual é a vantagem de gerar os arquivos de código objeto então?
Em projetos grandes, se uma modificação ocorrer, podemos apenas **recompilar o arquivo que foi modificado** e não o programa inteiro!

Arquivos de Programas

Finalmente, após o processamento do ligador sobre arquivos de código-objeto, é gerado um **arquivo executável**!

Note que, sejam arquivos de código-objeto ou arquivos executáveis, estes não são “preparados” para leitura humana. Ou seja, visualizadores/editores de texto não lidam bem com a sua exibição.

Porém, **a aplicação *hexdump* é capaz de exibir os bytes gravados** em tais arquivos.

Arquivos de Programas

ARQUIVO EXECUTÁVEL

```
$ hexdump -C programa
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  03 00 3e 00 01 00 00 00  60 10 00 00 00 00 00 00  |..>.....`.......|
00000020  40 00 00 00 00 00 00 00  98 36 00 00 00 00 00 00  |@........6......|
00000030  00 00 00 00 40 00 38 00  0d 00 40 00 1f 00 1e 00  |....@.8...@.....|
00000040  06 00 00 00 04 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000050  40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |@.......@.......|
00000060  d8 02 00 00 00 00 00 00  d8 02 00 00 00 00 00 00  |................|
00000070  08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  |................|
00000080  18 03 00 00 00 00 00 00  18 03 00 00 00 00 00 00  |................|
00000090  18 03 00 00 00 00 00 00  1c 00 00 00 00 00 00 00  |................|
000000a0  1c 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
000000b0  01 00 00 00 04 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  28 06 00 00 00 00 00 00  28 06 00 00 00 00 00 00  |(.......(.......|
000000e0  00 10 00 00 00 00 00 00  01 00 00 00 05 00 00 00  |................|
000000f0  00 10 00 00 00 00 00 00  00 10 00 00 00 00 00 00  |................|
00000100  00 10 00 00 00 00 00 00  81 01 00 00 00 00 00 00  |................|
00000110  81 01 00 00 00 00 00 00  00 10 00 00 00 00 00 00  |................|
00000120  01 00 00 00 04 00 00 00  00 20 00 00 00 00 00 00  |......... ......|
00000130  00 20 00 00 00 00 00 00  00 20 00 00 00 00 00 00  |. ....... ......|
00000140  f4 00 00 00 00 00 00 00  f4 00 00 00 00 00 00 00  |................|
00000150  00 10 00 00 00 00 00 00  01 00 00 00 06 00 00 00  |................|
00000160  b8 2d 00 00 00 00 00 00  b8 3d 00 00 00 00 00 00  |.-.......=......|
```

Mas, mesmo analisando o conteúdo do arquivo executável, não conseguimos ter ampla clareza do seu conteúdo.

Poucas informações podem ser diretamente interpretadas por humanos!

Arquivos de Programas

```
$ objdump -s programa

programa: formato do arquivo elf64-x86-64

Conteúdo da seção .interp:
 0318 2f6c6962 36342f6c 642d6c69 6e75782d  /lib64/ld-linux-
 0328 7838362d 36342e73 6f2e3200           x86-64.so.2.
Conteúdo da seção .note.gnu.property:
 0338 04000000 20000000 05000000 474e5500  .... .......GNU.
 0348 020000c0 04000000 03000000 00000000  ................
 0358 028000c0 04000000 01000000 00000000  ................
Conteúdo da seção .note.gnu.build-id:
 0368 04000000 14000000 03000000 474e5500  ............GNU.
 0378 59bb8b95 21a1e798 23840ece b9c5d7e3  Y...!...#.......
 0388 00405634                             .@V4
...
Conteúdo da seção .fini:
 1174 f30f1efa 4883ec08 4883c408 c3        ....H...H....
Conteúdo da seção .rodata:
 2000 01000200 48656c6c 6f20576f 726c6400  ....Hello World.
Conteúdo da seção .eh_frame_hdr:
 2010 011b033b 34000000 05000000 10f0ffff  ...;4...........
 2020 68000000 30f0ffff 90000000 40f0ffff  h...0.......@...
 2030 a8000000 50f0ffff 50000000 39f1ffff  ....P...P...9...
 2040 c0000000                             ....
Conteúdo da seção .eh_frame:
 2048 14000000 00000000 017a5200 01781001  .........zR..x..
 2058 1b0c0708 90010000 14000000 1c000000  ................
 2068 f8efffff 26000000 00440710 00000000  ....&....D......
 2078 24000000 34000000 a0efffff 20000000  $...4....... ...
 2088 000e1046 0e184a0f 0b770880 003f1a3a  ...F..J..w...?.:
```

ARQUIVO EXECUTÁVEL

Outra opção de aplicação para analisar o arquivo executável é a aplicação *objdump*.

O *objdump* irá traduzir algumas informações do arquivo executável e apresentar as suas seções.

Arquivos de Programas

ARQUIVO EXECUTÁVEL

A aplicação objdump apresenta arquivos executáveis divididos em **vinte e seis (26) seções diferentes**.

Ao longo da disciplina, estudaremos características dessas seções, compreendendo o que cada uma representa na formação de um programa na totalidade.

Arquivos de Programas

ARQUIVO: EXECUTÁVEL Vs. OBJETO

```
$ objdump -s programa.o

programa.o: formato do arquivo elf64-x86-64

Conteúdo da seção .text:
 0000 f30f1efa 554889e5 4883ec10 897dfc48  ....UH..H....}.H
 0010 8975f048 8d050000 00004889 c7e80000  .u.H......H.....
 0020 0000b800 000000c9 c3                 .........
Conteúdo da seção .rodata:
 0000 48656c6c 6f20576f 726c6400           Hello World.
Conteúdo da seção .comment:
 0000 00474343 3a202855 62756e74 75203131  .GCC: (Ubuntu 11
 0010 2e332e30 2d317562 756e7475 317e3232  .3.0-1ubuntu1~22
 0020 2e30342e 31292031 312e332e 3000      .04.1) 11.3.0.
Conteúdo da seção .note.gnu.property:
 0000 04000000 10000000 05000000 474e5500  ............GNU.
 0010 020000c0 04000000 03000000 00000000  ................
Conteúdo da seção .eh_frame:
 0000 14000000 00000000 017a5200 01781001  .........zR..x..
 0010 1b0c0708 90010000 1c000000 1c000000  ................
 0020 00000000 29000000 00450e10 8602430d  ....)....E....C.
 0030 06600c07 08000000                    .`......
```

Ao analisarmos o arquivo objeto com o objdump, entretanto, apenas cinco (5) seções são listadas.

Note a escalada de complexidade entre o arquivo de código-objeto e o executável.

Arquivos Executáveis

PORÉM, A ORGANIZAÇÃO E OS DADOS DE UM ARQUIVO EXECUTÁVEL SERÁ SEMPRE IGUAL EM QUALQUER CIRCUNSTÂNCIA?

Arquivos Executáveis

NÃO! Os arquivos executáveis dependem do sistema computacional onde os quais irão executar, de maneira bem geral:

- As operações dependem da arquitetura do sistema
- O formato depende do sistema operacional

Arquivos Executáveis

As operações (**AMD64**) serão nosso objeto de estudo nas próximas aulas!

E quanto aos formatos?

- Windows: COM; EXE; PE
- MAC: PEF
- Android: DEX
- Linux: ELF

De maneira geral, os formatos são semelhantes, mas não intercambiáveis.

Arquivos Executáveis

No contexto da nossa disciplina, o formato estudado será apenas o ELF (*Executable and Linkable Format*).

O ELF é uma evolução do formato COFF (*Common Object File Format*), com especificação aberta (https://refspecs.linuxbase.org/elf/elf.pdf).

O ELF é adotado no Linux desde 1999!

Programa Vs. Processo

Um ponto é termos um programa como um arquivo executável; outra é termos um processo (programa em execução).

Para que um programa passe a ser um processo, o sistema operacional, através do carregador, deve **ler o arquivo executável e mapeá-lo em um espaço de endereçamento do processo**.

Processos

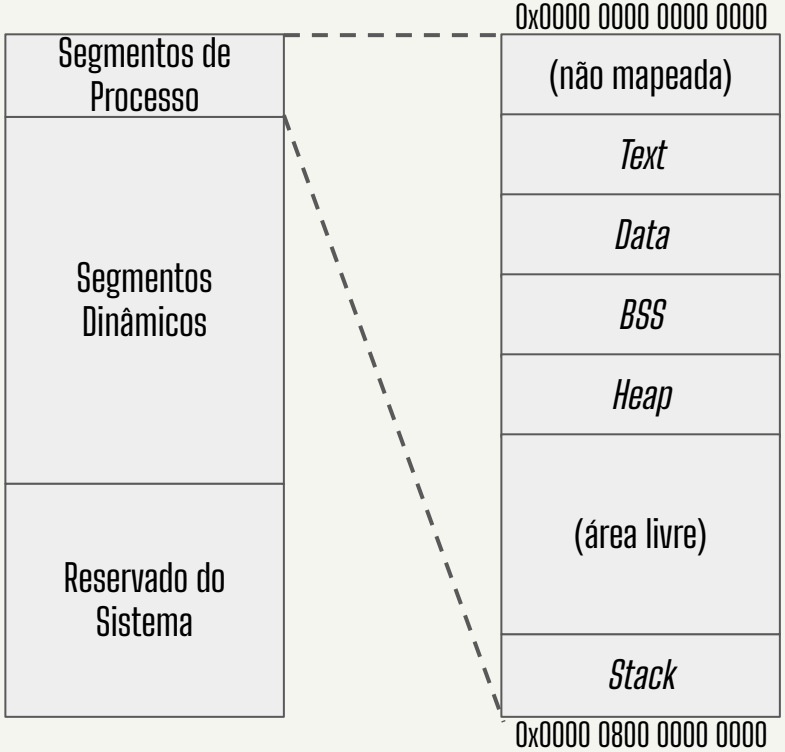
Quando um processo é criado, o sistema operacional fornece um espaço de endereçamento virtual para o mesmo.

O espaço contempla 2^{64} endereços

Processos

- Segmentos do Processo: utilizado para a alocação da imagem do processo (ELF)
- Segmentos dinâmicos: local onde objetos dinâmicos são alocados
- Reservado do Sistema: residência do SO e de objetos inacessíveis ao processo

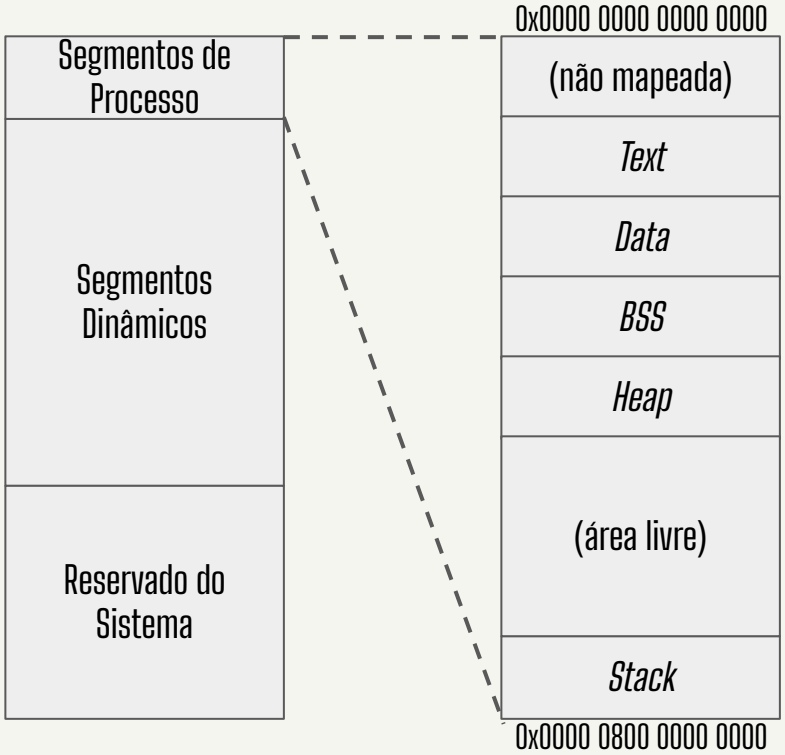
Processos



Algumas seções do arquivo executável são carregadas nos segmentos de processo, as principais são:

- *Text*: código
- *Data*: dados globais inicializados
- *BSS*: dados não inicializados
- *Heap*: variáveis dinâmicas
- *Stack*: pilha de execução

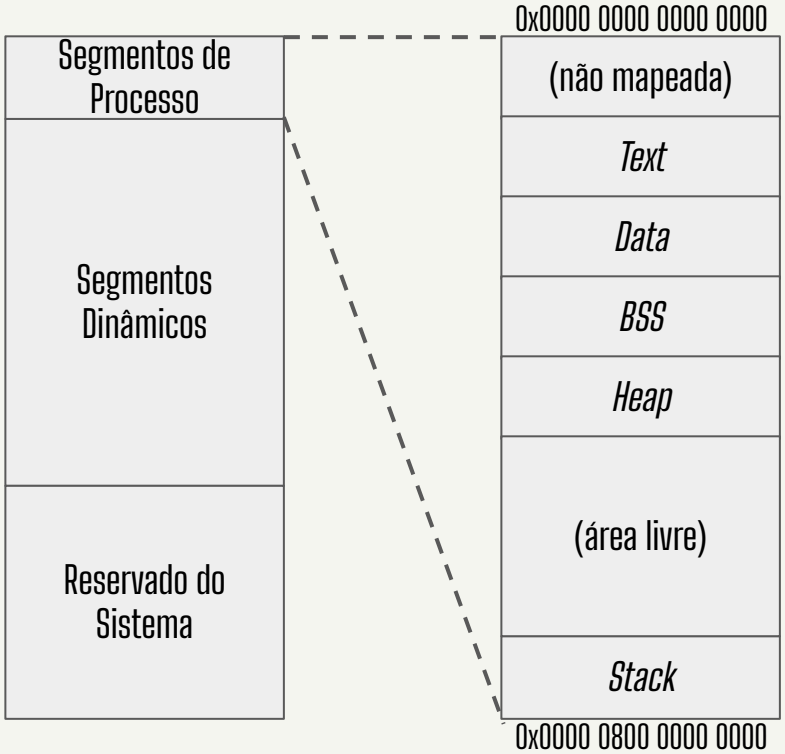
Processos



- Então, cada processo tem 2^{64} endereços
  - 16 EiB
- Considerando a área do processo, há 2^{48} endereços
  - 256 TiB
- Para endereçar 16GB precisamos apenas 2^{34} endereços

O que isso quer dizer?

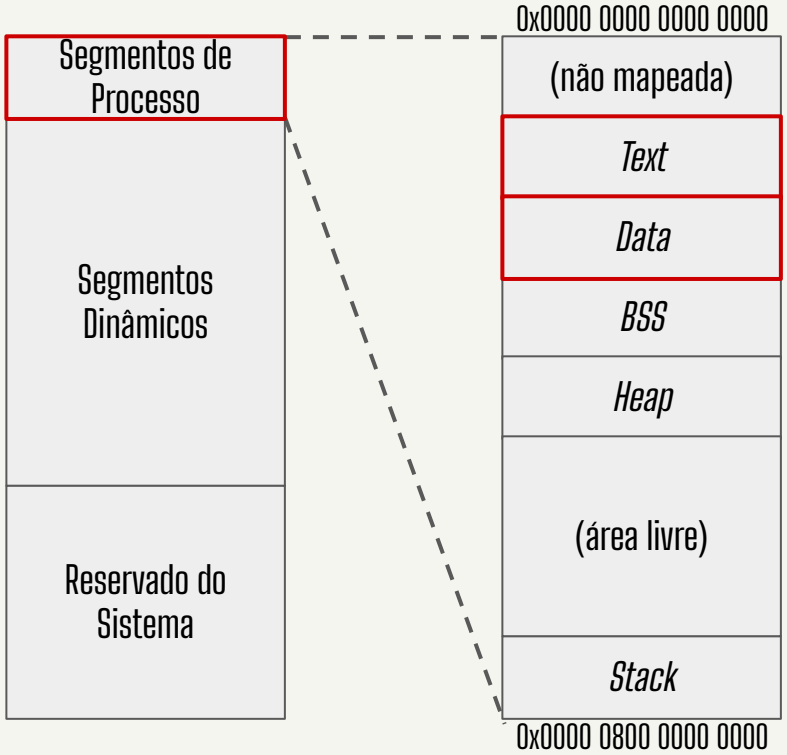
Processos



Sabendo que o espaço de endereçamento é maior que a memória física disponível, como fechar a conta?

Memória Virtual (Paginação)

Processos

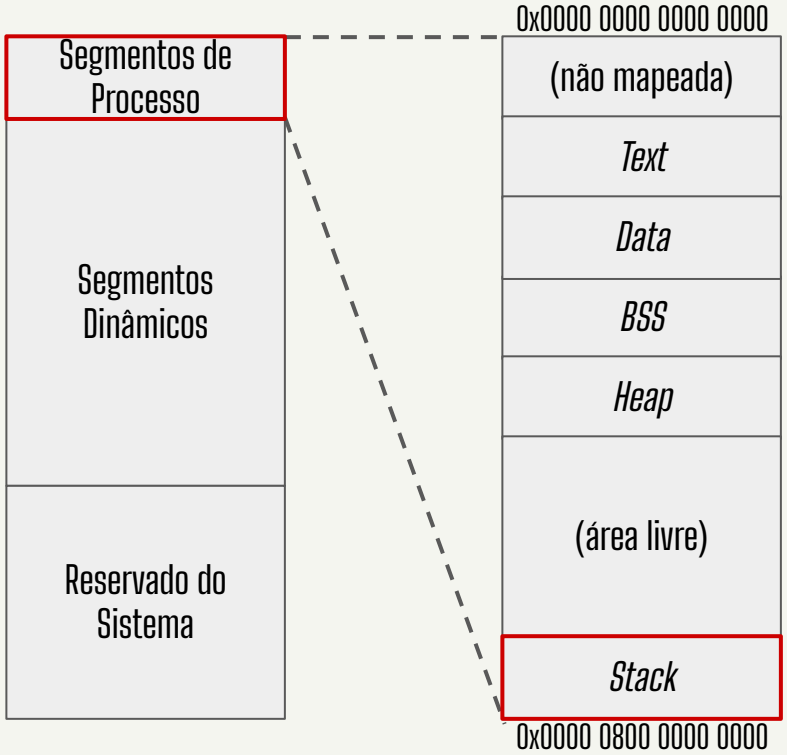


E os próximos passos?

Já nas próximas aulas, vamos **estudar a seção *text* e *data*** via código *assembly*.

- Atribuições
- Condicionais
- Repetições
- ...

Processos

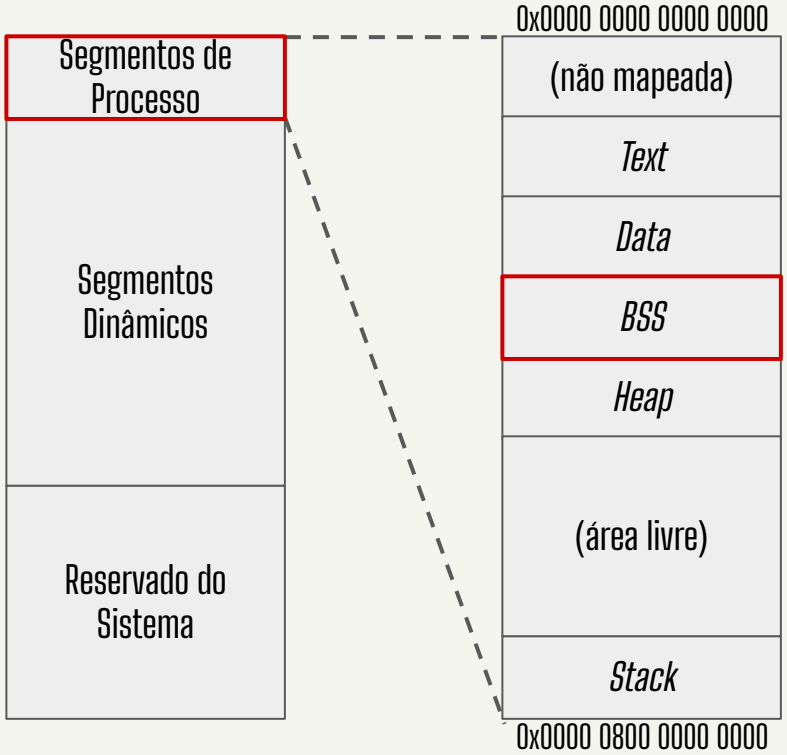


E os próximos passos?

Em seguida, seguindo a mesma abordagem, **estudaremos a seção *stack***.

- Procedimentos
- Parâmetros

Processos

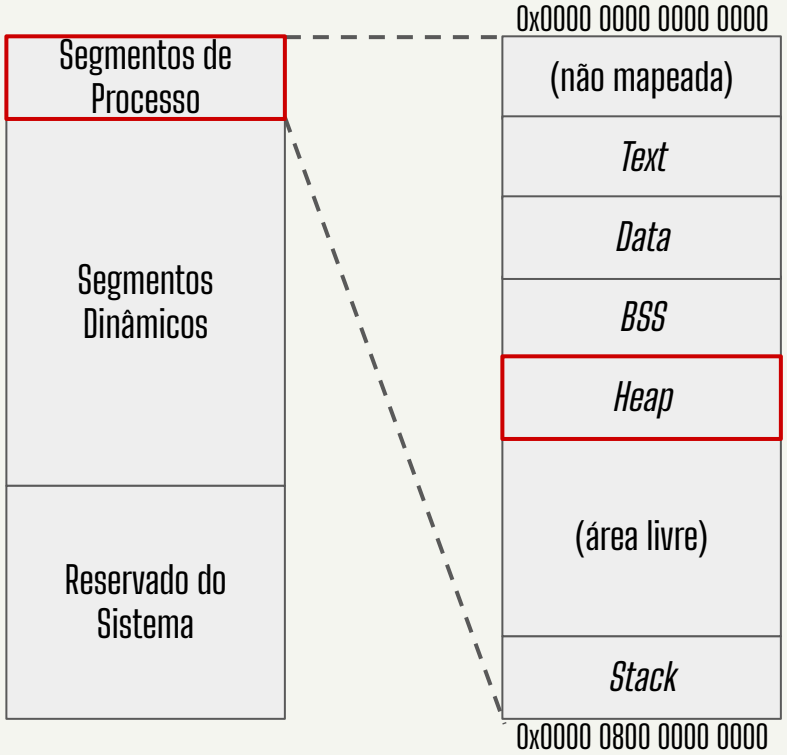


E os próximos passos?

Então, o objetivo torna-se **estudar a seção *BSS***.

- Variáveis não inicializadas
- Chamadas de sistema

Processos

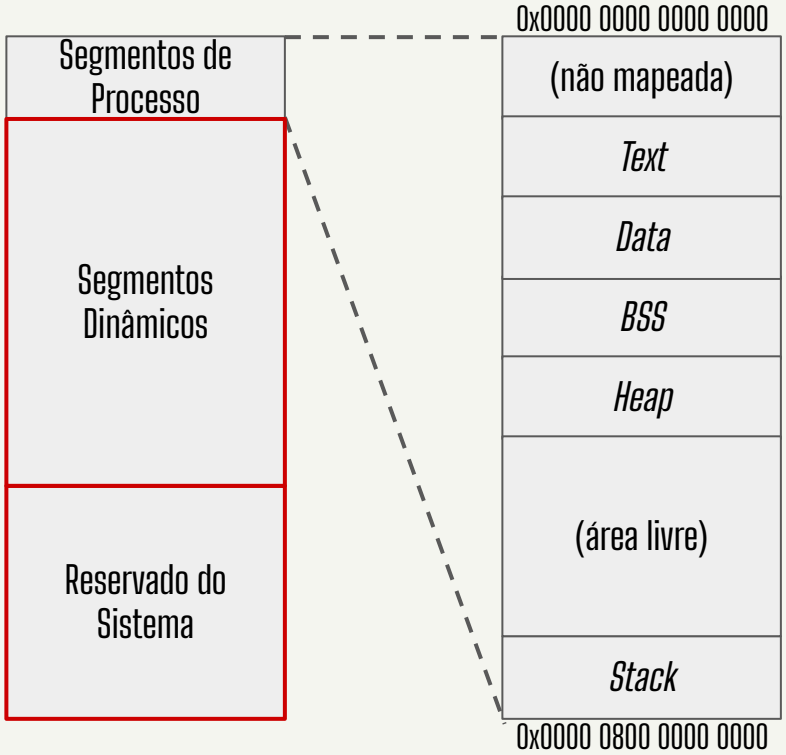


E os próximos passos?

Finalmente, para encerrar as seções do segmento de processo, **vamos analisar a *heap***.

- Alocação dinâmica de memória

Processos

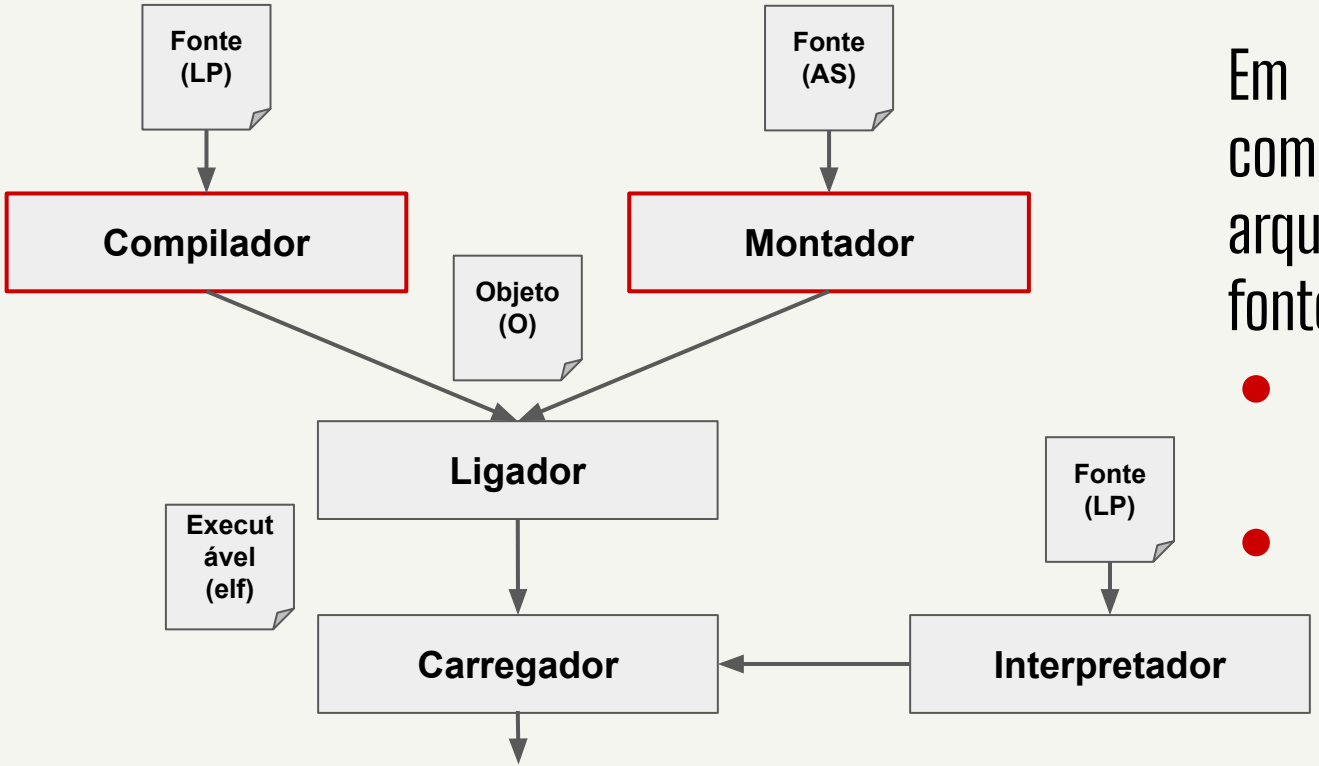


E os próximos passos?

Então, na segunda parte da disciplina, **o foco será os segmentos dinâmicos e reservados do sistema.**

- Ligadores
- Carregadores
- Interpretadores

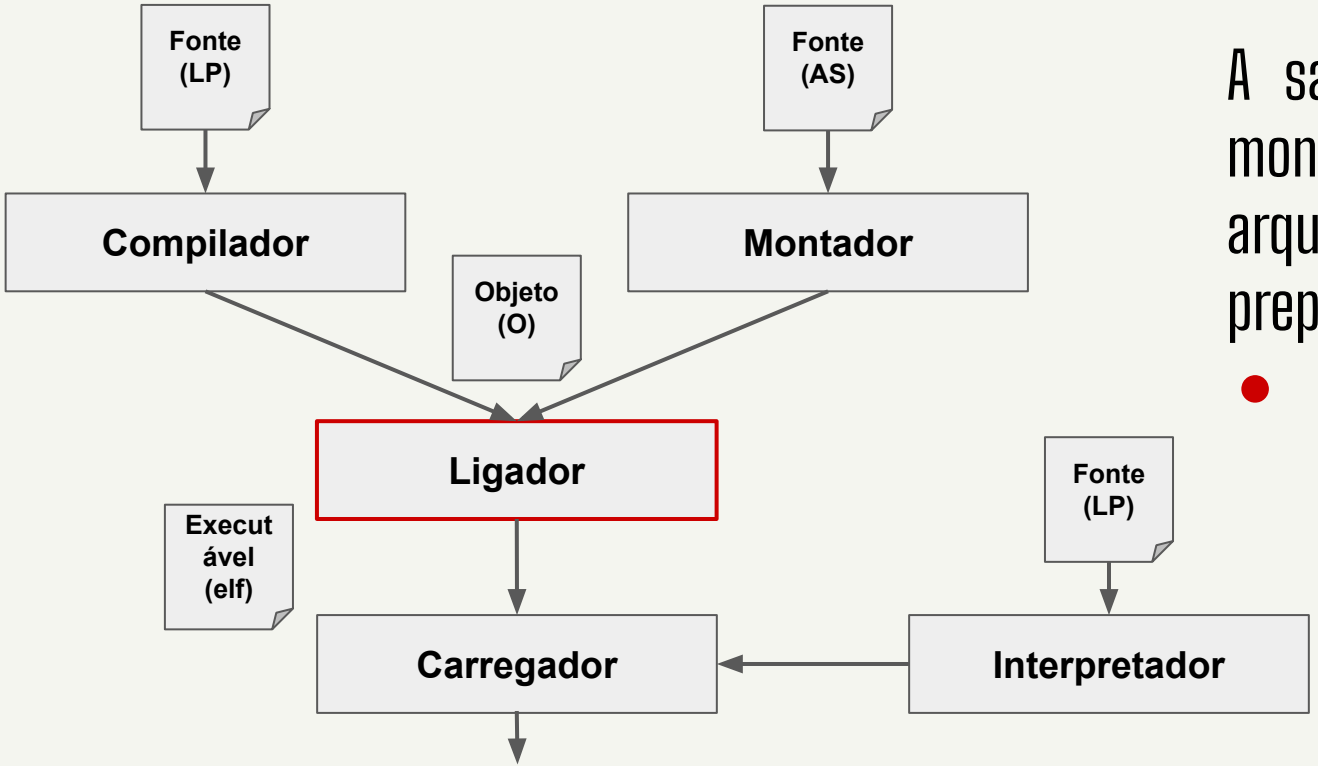
Do Fonte ao Processo



Em um processo de compilação, o primeiro arquivo tipicamente é o fonte:

- Compilador (linguagem "tradicional")
- Montador (*assembly*)

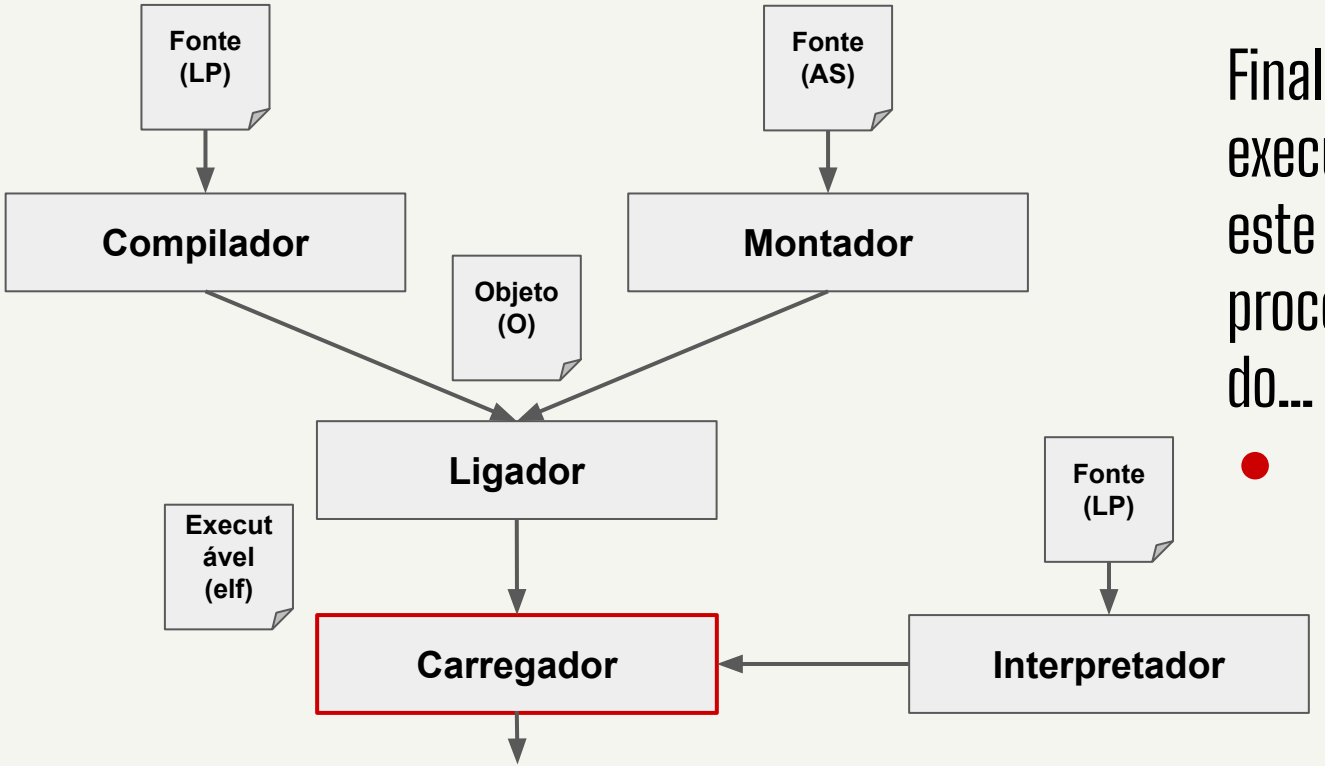
Do Fonte ao Processo



A saída do compilador ou montador é um (ou mais) arquivo objeto, ainda não preparado para a execução:

- Ligador

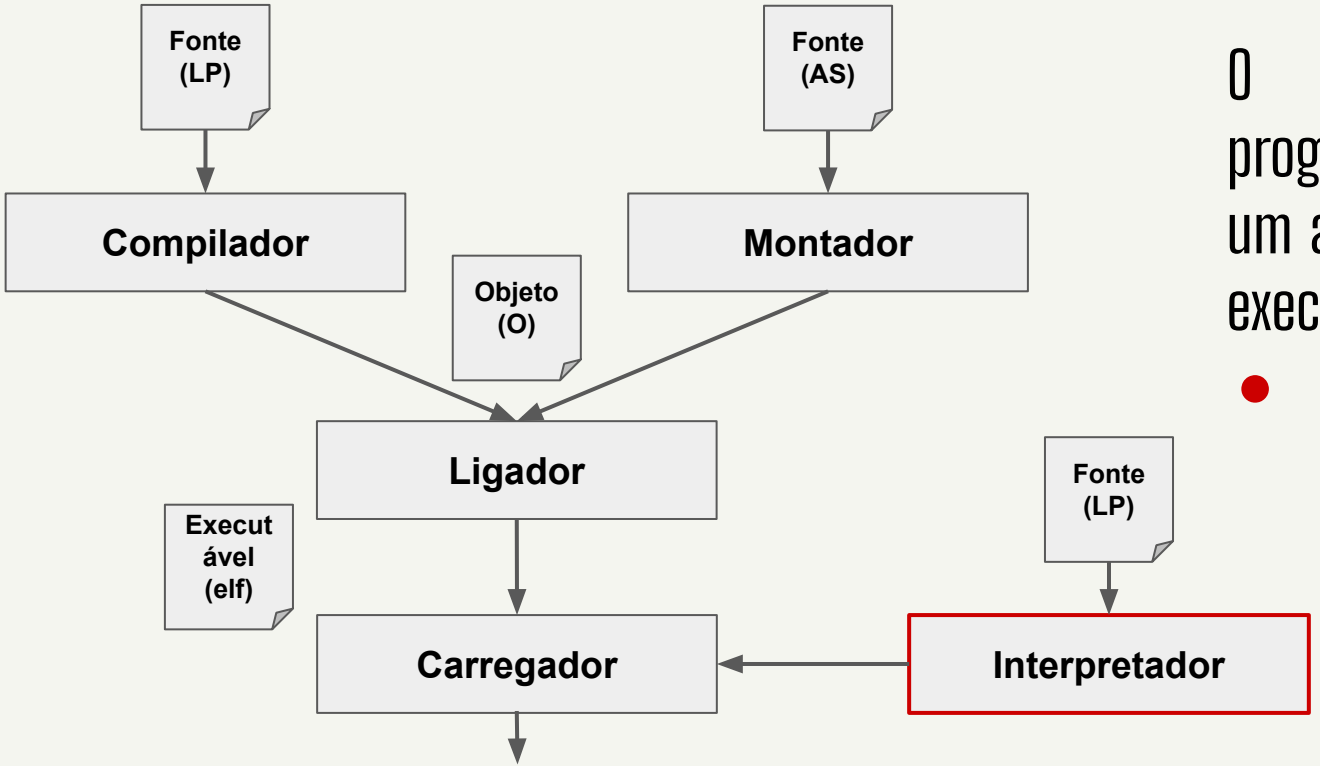
Do Fonte ao Processo



Finalmente o arquivo executável é gerado, sendo este transformado em um processo através da atuação do...

- Carregador

Do Fonte ao Processo



O interpretador é um programa executável que lê um arquivo fonte e simula a execução do programa:

- Interpretador

Exercício #01

Vamos analisar cada tipo de arquivo de um programa simples. Porém, este programa deve ser o mais autocontido possível, para facilitar a visualização das estruturas desejadas.

Qual é o programa em C mais simples que vocês podem imaginar?

Após desenvolvermos o programa, vamos analisar os arquivos...

(i) Fonte **(ii)** *Assembly* **(iii)** Objeto **(iv)** Executável

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius/
viniciusfulber@ufpr.br