Anatomia dos Processos

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Março, 2017

Sumário

Pilha

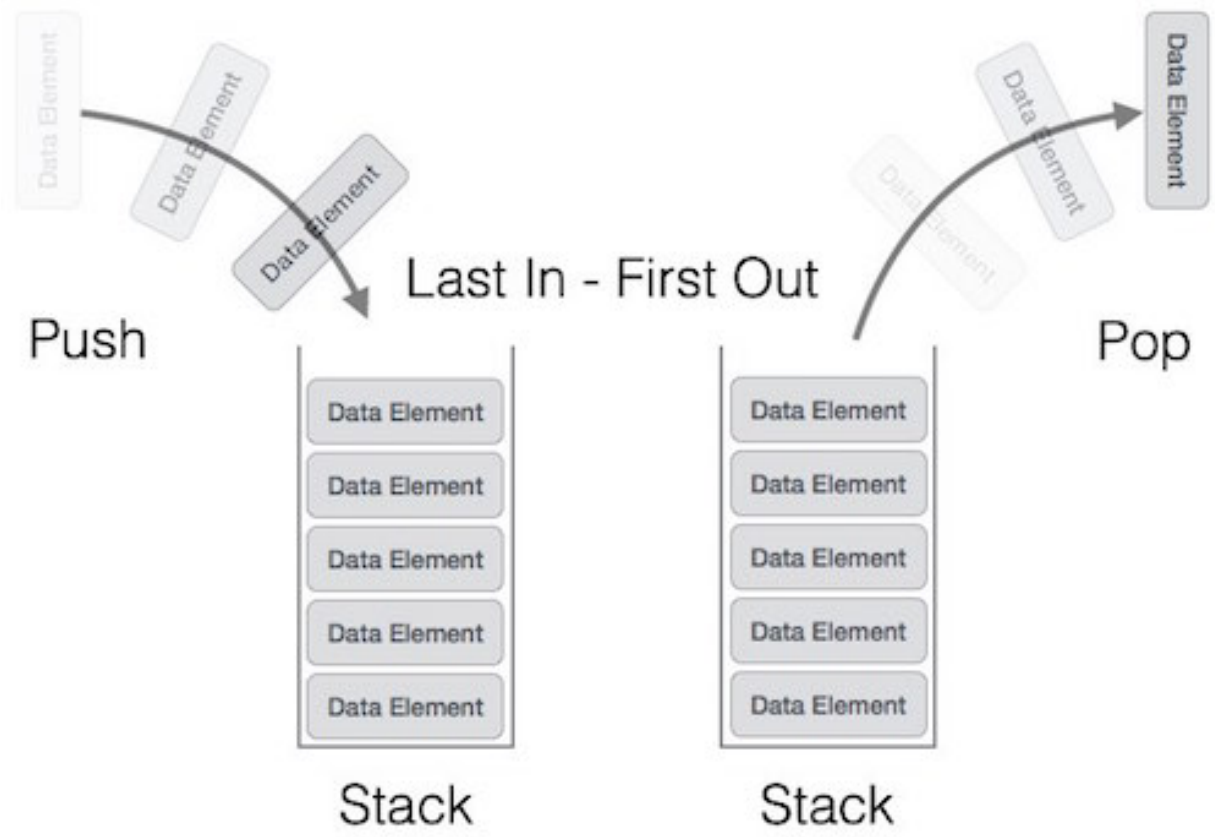


Figura 1.1: Representação de uma pilha [1]

[1] http://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm

Utilização

- Armazenamento do contexto;
- Último valor acessado;
- Ordem de execução.
- Ex.: Cálculo fatorial

$$0! = 1 \tag{1}$$
$$1! = 1 * 0! \tag{2}$$
$$2! = 2 * 1! \tag{3}$$
$$3! = 3 * 2! \tag{4}$$
$$4! = 4 * 3! \tag{5}$$
$$\ldots \tag{6}$$
$$n! = n * (n-1)! \tag{7}$$

Exemplo: fatorial

Listing 1: Cálculo fatorial recursivo

```
float fat(int n) {
    int x; /* define uma ávarivel local x */
    x=n-1;

    if (n == 0) return (1);
    else return (n * fat(x)); /* ao éinvs de passar n-1, passa-se x */
}

int main() {
    int i=0;

    while (i>=0){
        printf("Entre valor inteiro >=0 ou valor <0 para terminar:");
        scanf("%d", &i);
        if (i>=0) printf("Fatorial de %d = %f\n", i, fat(i));
    }

    return 0;
}
```

Listing 1: Cálculo fatorial recursivo

Heap

- Tipo de árvore;
- A chave na raiz é maior do que em qualquer um dos filhos;
- Ambas subárvores (direita e esquerda) também são *heaps*;
- Utilizado para fila de prioridades;
- Método de Ordenação eficiente: $O(nlog(n))$

Heap

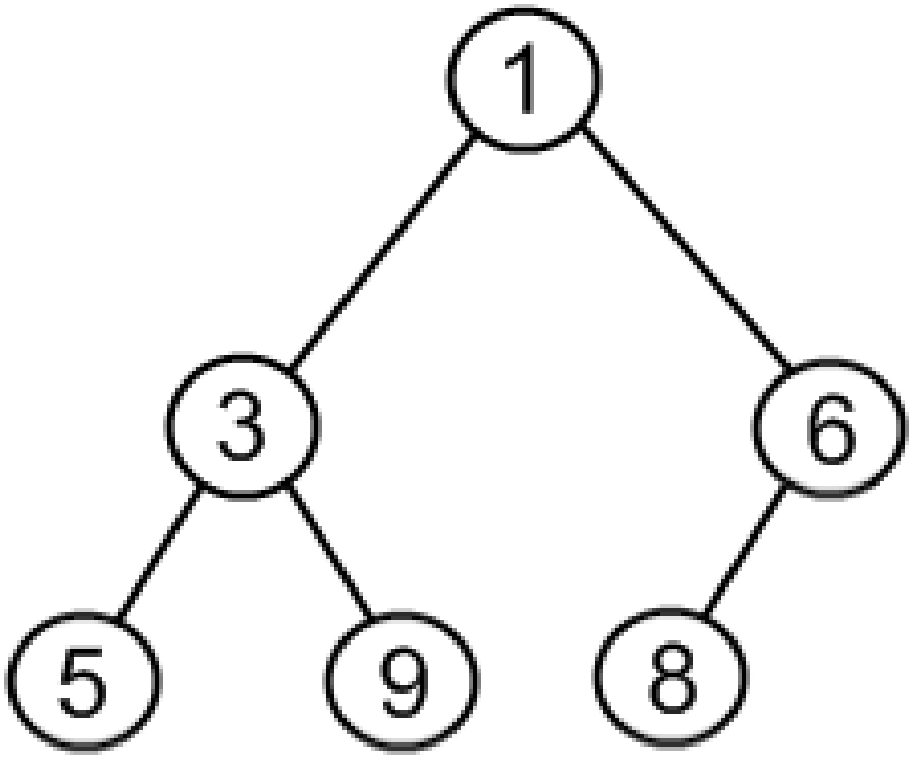


Figura 1.2: Estrutura de Heap [Algolist, 2016]

Inserção

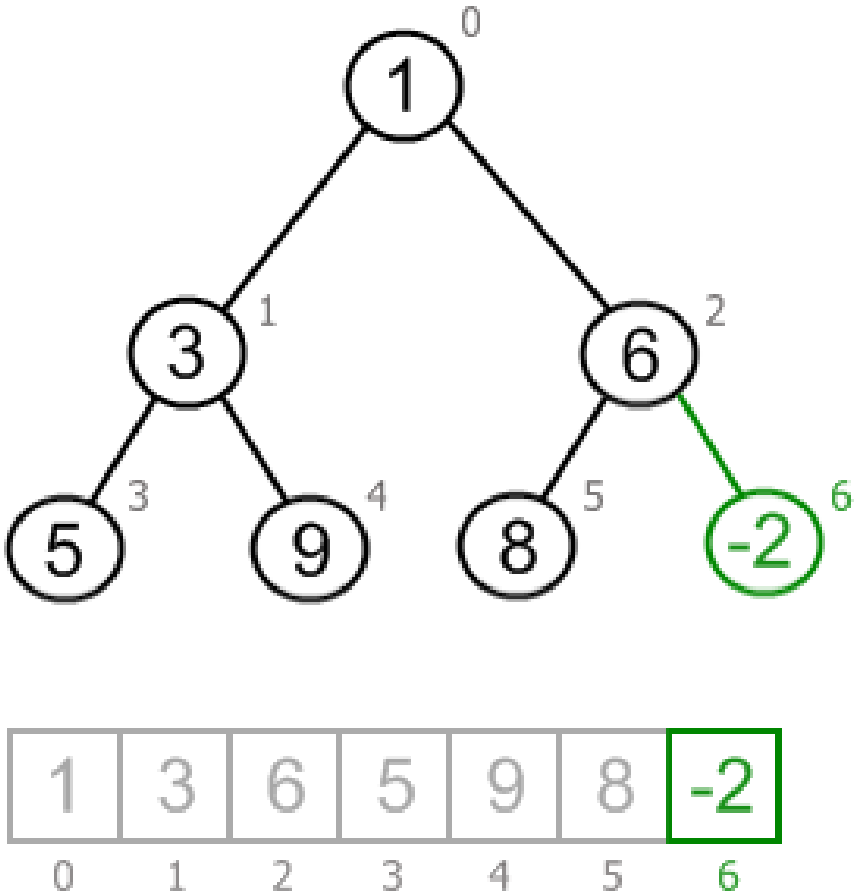


Figura 1.3: Inserção do elemento -2 [Algolist, 2016]

Propriedades da heap

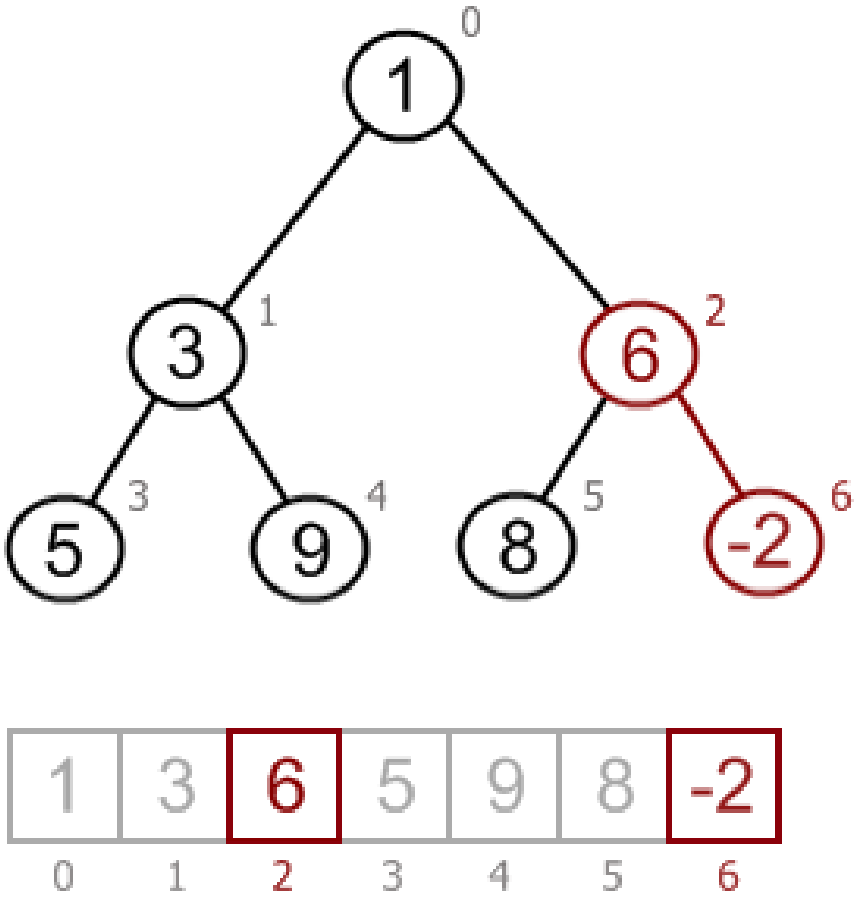


Figura 1.4: Propriedade da heap quebrada (menor no topo) [Algolist, 2016]

Reordenação

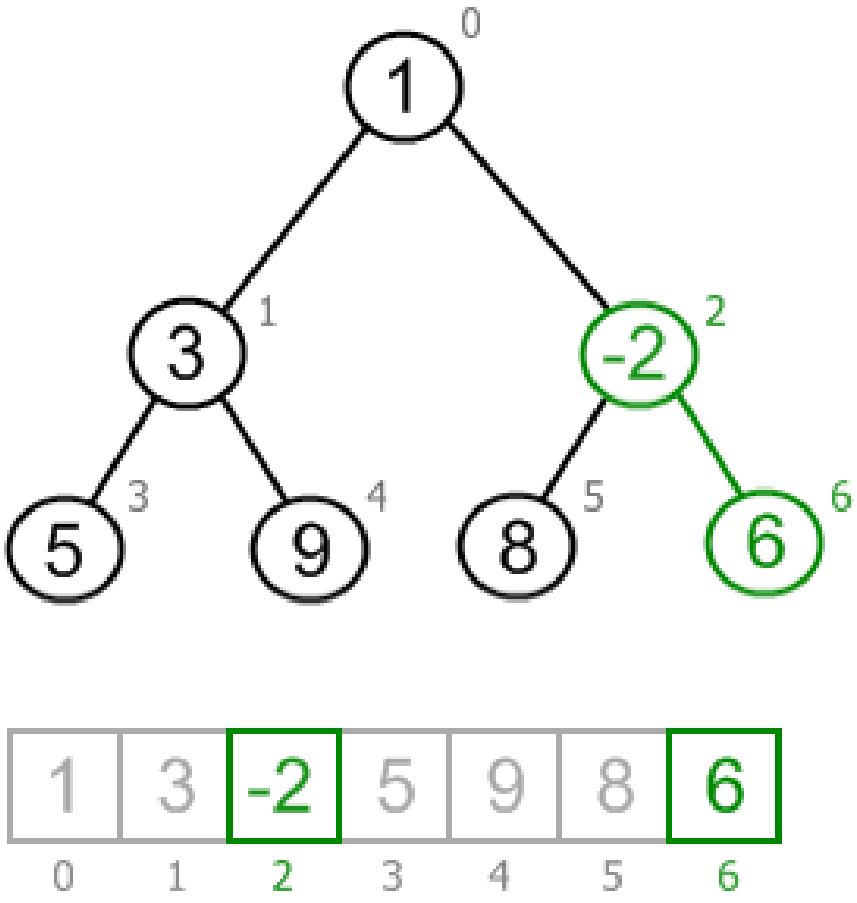


Figura 1.5: Para manter as propriedades, há uma reordenação da heap [Algolist, 2016]

Propriedade da raiz

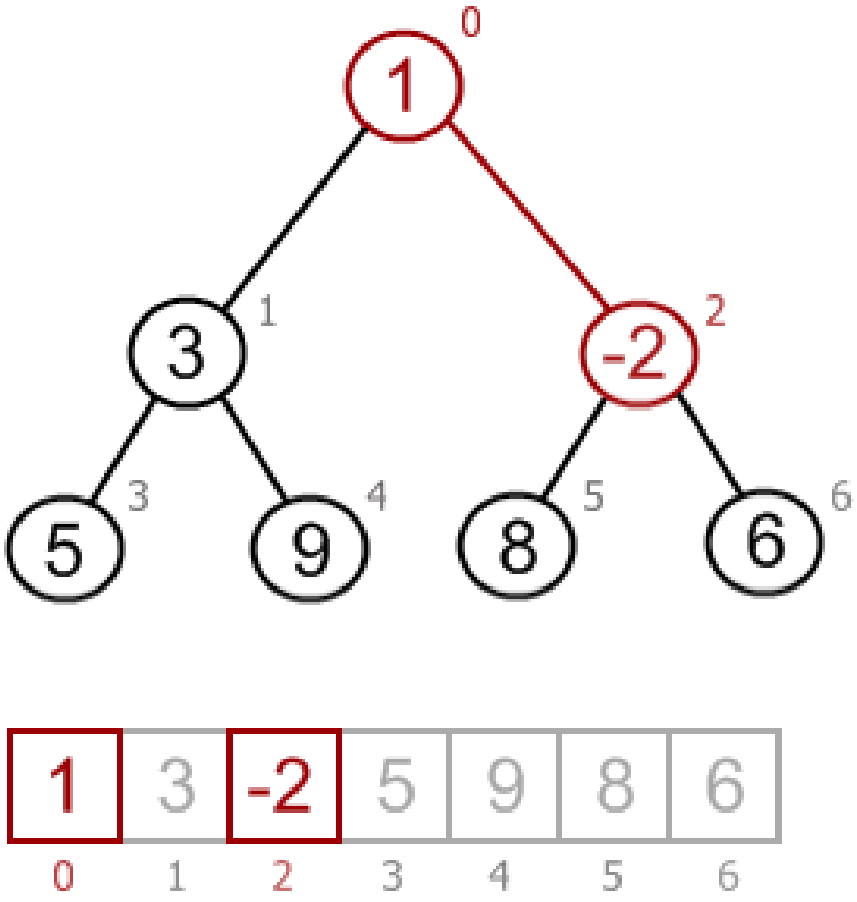


Figura 1.6: Propriedade quebrada na raiz [Algolist, 2016]

Nova reordenação

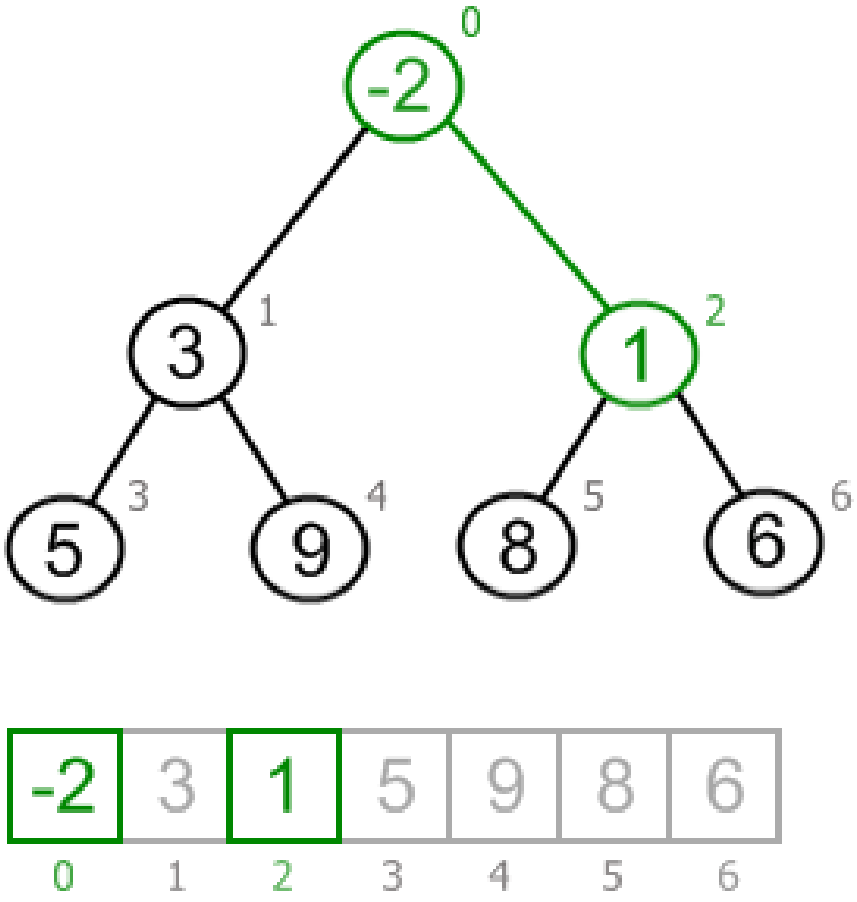


Figura 1.7: Nova reordenação para manter as propriedades [Algolist, 2016]

Processos x Tarefas

Processos são programas que estão sendo executados em um espaço virtual de endereçamento exclusivo.

- Em sistemas Unix, processos são estruturas de dados que contém informações necessárias para a execução do programa, como conteúdo dos registradores e memória;
- Princípio básico: separar a operação de criação de um processo da operação de execução de um programa;
- São separadas por chamadas de função diferentes: *fork()* e *exec()*;
  - *fork()* cria um processo que tem como pai o processo que a chamou;
  - *exec()* cria um novo programa como uma sequência de processos, que tem como pai o contexto de execução, muitas vezes o próprio init.

Elementos do processo

Contexto de hardware Conteúdo dos registradores.

- Sistemas de tempo compartilhado: dividem o uso do processador.
- Mudança de contexto.

Contexto de software Características em tempo de execução

- Identificação: PID e UID
- Quotas: memória, E/S, processos filhos e threads
- Privilégios

Espaço de endereçamento Espaço de memória utilizada pelo processo

Estados dos processos (Gráfico)

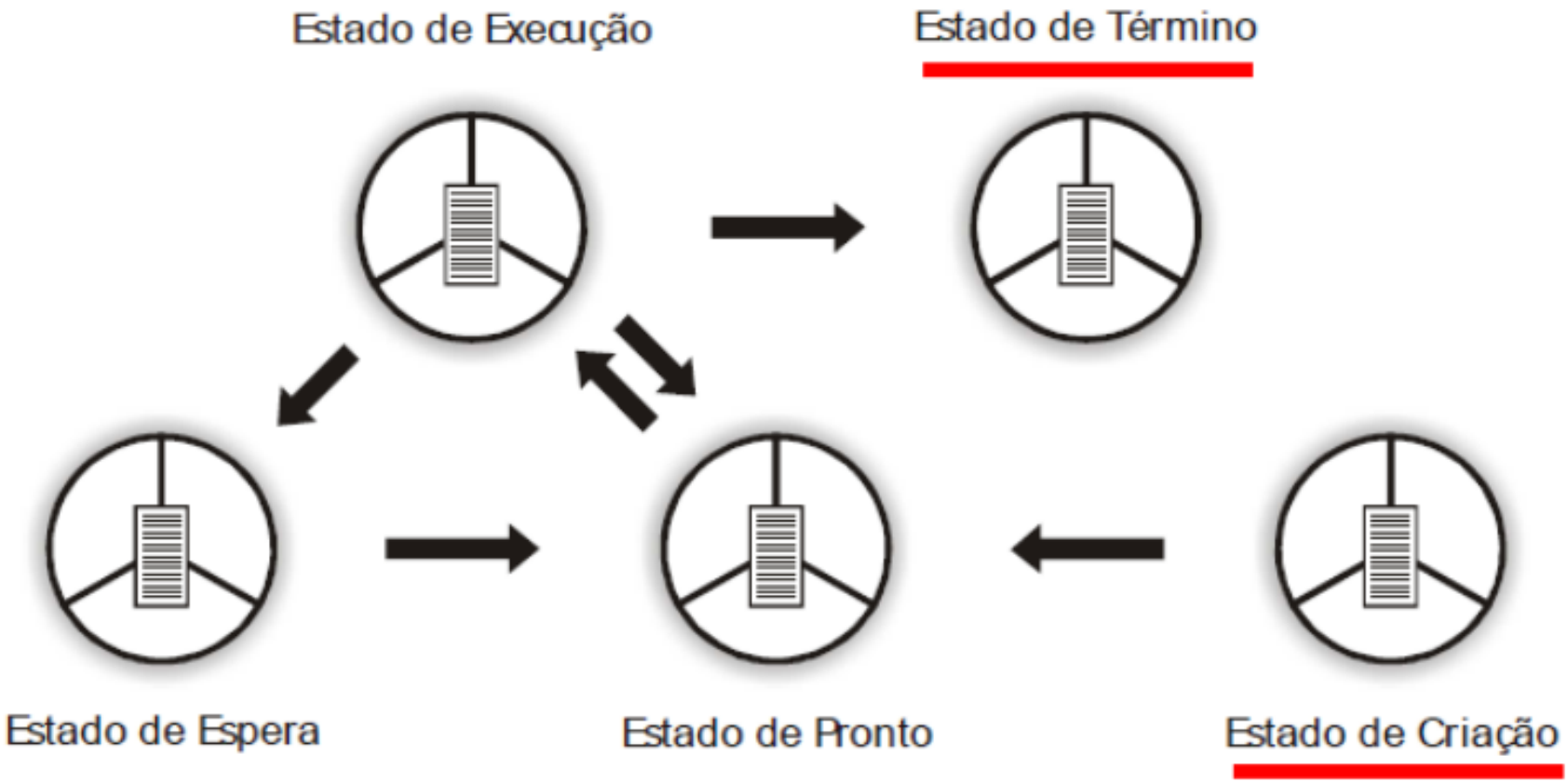


Figura 2.1: Estados dos processos [Chagas, 2016]

Troca de contexto

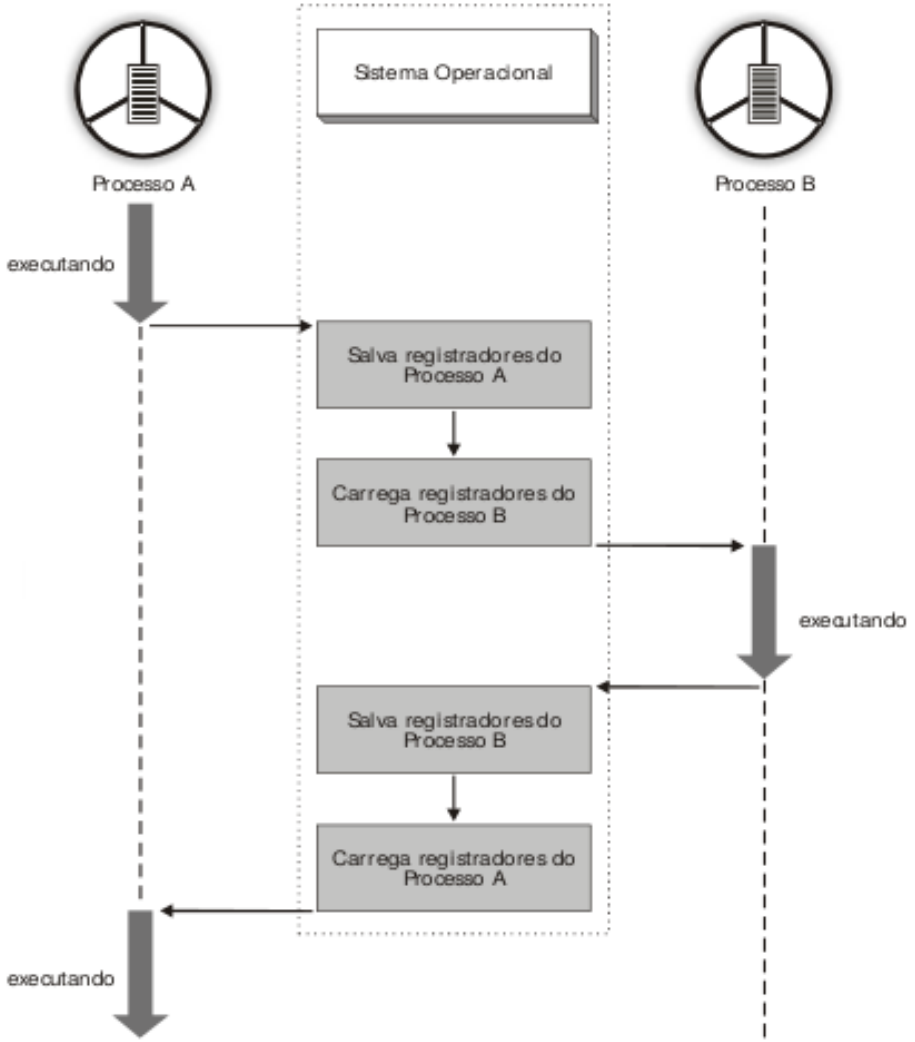


Figura 2.2: Troca de contexto dos processos [Favacho, 2009]

Programação Síncrona

- Concorrência [Chagas, 2016]:
  1. O programa perde o uso do processador;
  2. O programa retorna para continuar o processamento;
  3. O estado do programa deve ser idêntico ao do momento em que foi interrompido.
- O programa continua a execução exatemente na instrução seguinte.

O modelo é dito síncrono porque as saídas do sistema podem ser vistas como sincronizadas com as suas entradas.
[FARINES and MELO, 2000]

Modelos Síncronos

Uma das características mais importantes encontrada nos modelos síncronos é a rejeição do não determinismo. [FARINES and MELO, 2000, p.105]

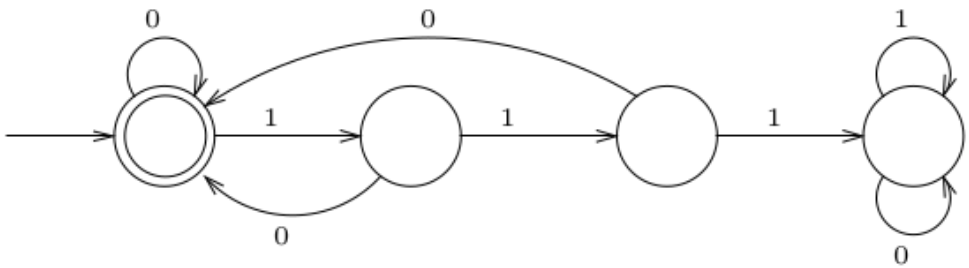
Determinismo Para cada estado do programa, existe somente uma possibilidade para a função de transição (próximo estado);

Não determinismo Podem existir várias escolhas para o próximo estado em qualquer ponto.

Determinismo

Determinístico

Exatamente uma trajetória sobre uma $w \in \Sigma^*$.



Não-determinístico

Nenhuma, uma ou várias trajetórias sobre uma $w \in \Sigma^*$.

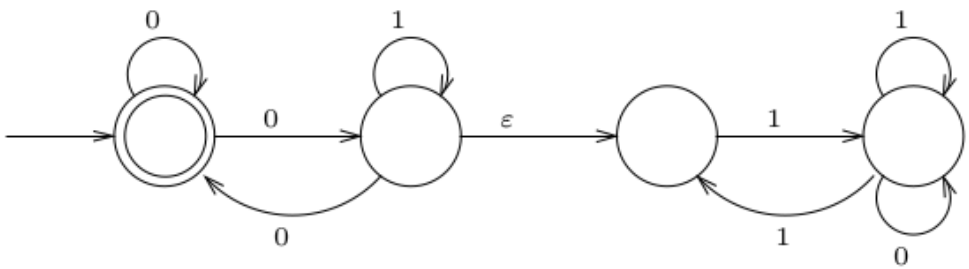


Figura 2.3: Autômatos Finitos Determinísticos e Não Determinísticos [Rezende, 2016]

Concorrência

- Paradigma: controlar/restringir o acesso ao recurso;
- O controle de acesso aos recursos é realizado através de eventos;
- Eventos inesperados pode causar um desvio inesperado no fluxo de execução.

Interrupção Realizada por algum evento externo ao programa, independente da instrução.

- Podem ser geradas por eventos assíncronos;
- Até várias vezes ao mesmo tempo.

Exceção Erro na instrução de algum programa. Ex.: falha de segmentação (*segfault*).

- Sempre gerada por um evento síncrono;
- Somente um evento de cada vez.

Interrupção

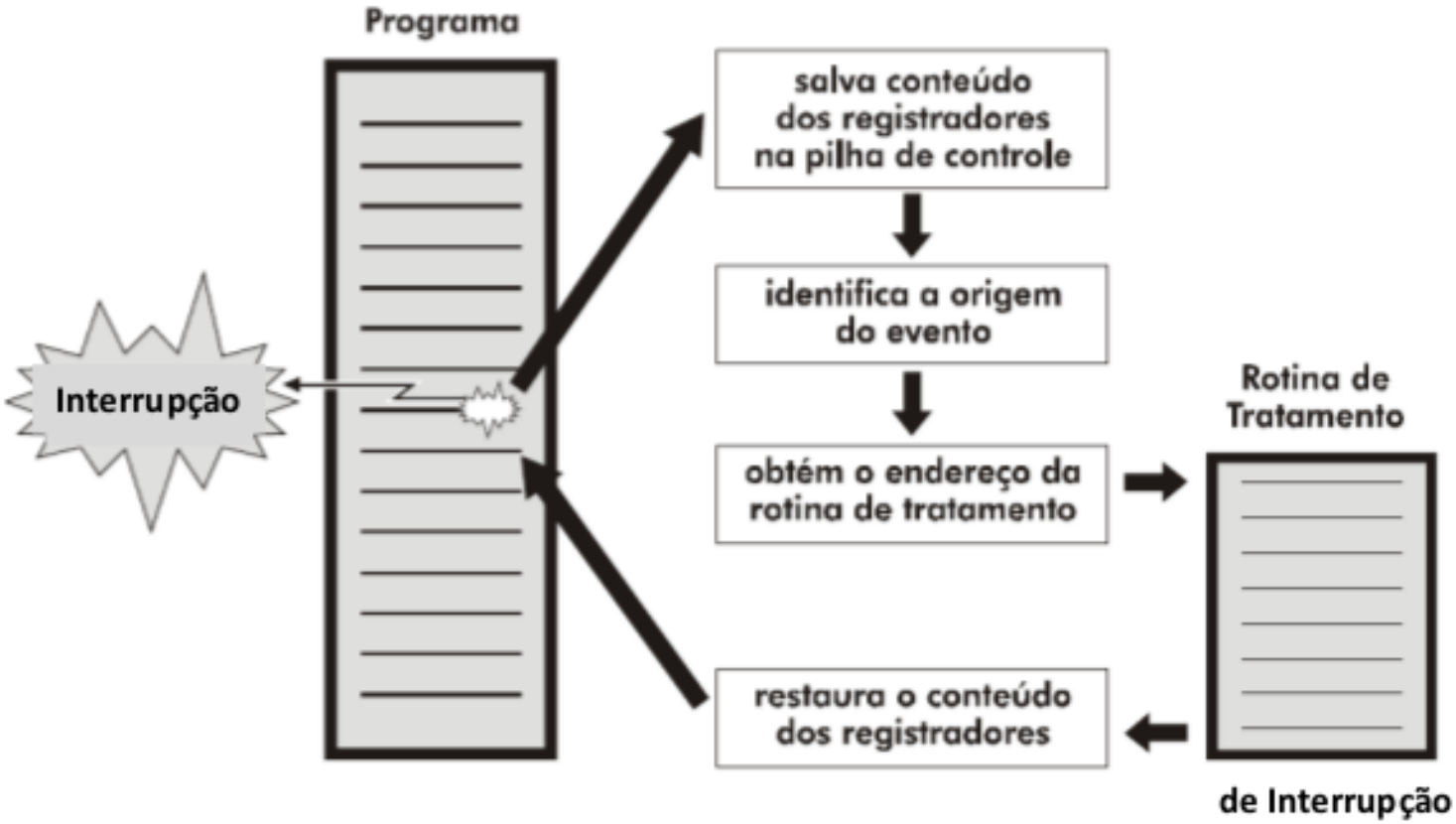


Figura 2.4: Tratamento de Interrupção [Chagas, 2016]

Exceção

Figura 2.5: Tratamento de Exceção [Rezende, 2016]

Estados dos processos

Durante o ciclo de vida de um processo ele passa por diferentes estados. Em sistemas Unix [Guarezi and Silva, 2010] são:

run Está sendo executado no processador;

ready ou executável Dispõe de todos os recursos que precisa e está pronto para ser executado;

sleep ou dormente Bloqueado à espera de algum recurso, e só pode ser desbloqueado se receber um sinal de outro processo;

zumbi Caso cada vez mais raro, onde um processo é criado por um programa, que por sua vez é finalizado antes de receber o resultado do processo;

parado Recebeu ordem do administrador para interromper a execução. Será reiniciado se receber um sinal de continuação (CONT).

Espaço Virtual de Memória

- No modo 32-bit é um bloco fixo de espaço;
- Paginação;
- Mantida pelo Sistema Operacional e consultada pelo processo.

Espaço Memória Virtual (32 bit)

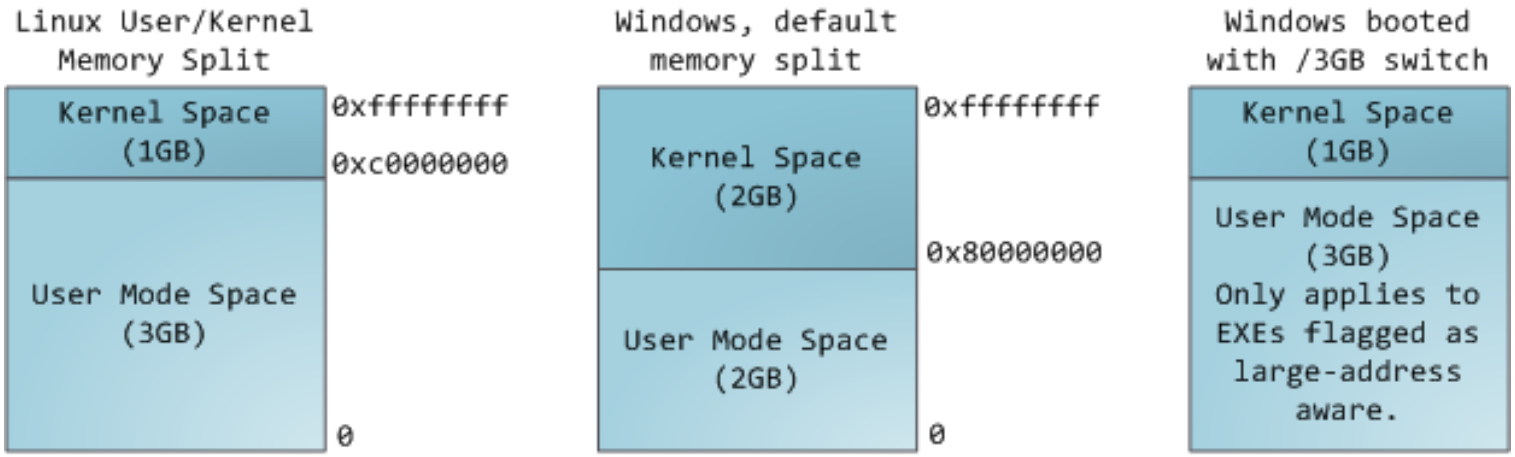


Figura 3.1: Utilização do endereço virtual de memória [Duarte, 2009]

Endereçamento

- O kernel sobe em um espaço de endereçamento exclusivo;
- No Linux, o mesmo espaço é mapeado em todos os processos;
- Interrupções e chamadas de sistema (SYSCALL);
- Troca de contexto muda o espaço de memória ativo.

Endereçamento

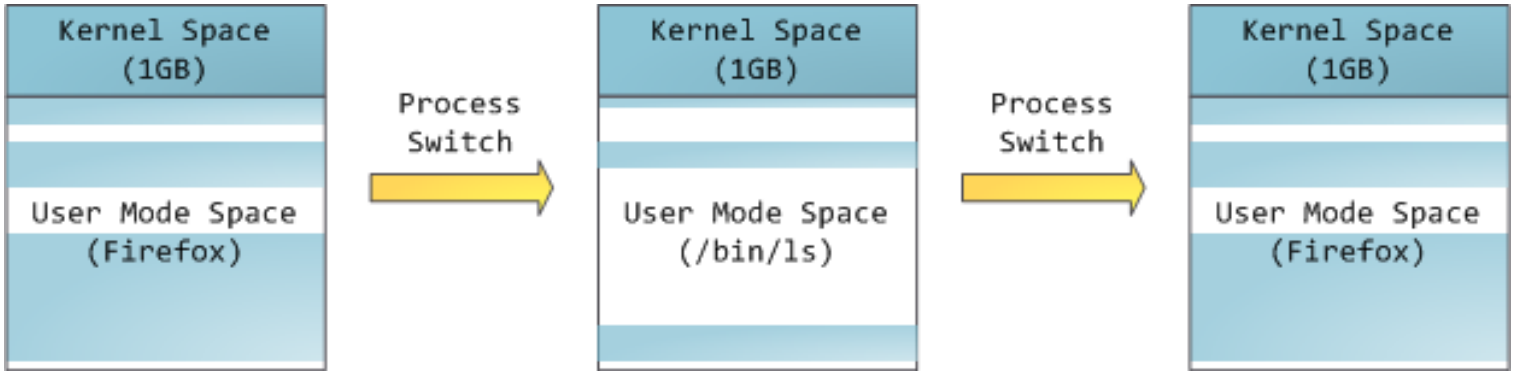


Figura 3.2: Troca de contexto entre processos [Duarte, 2009]

Segmentos de memória

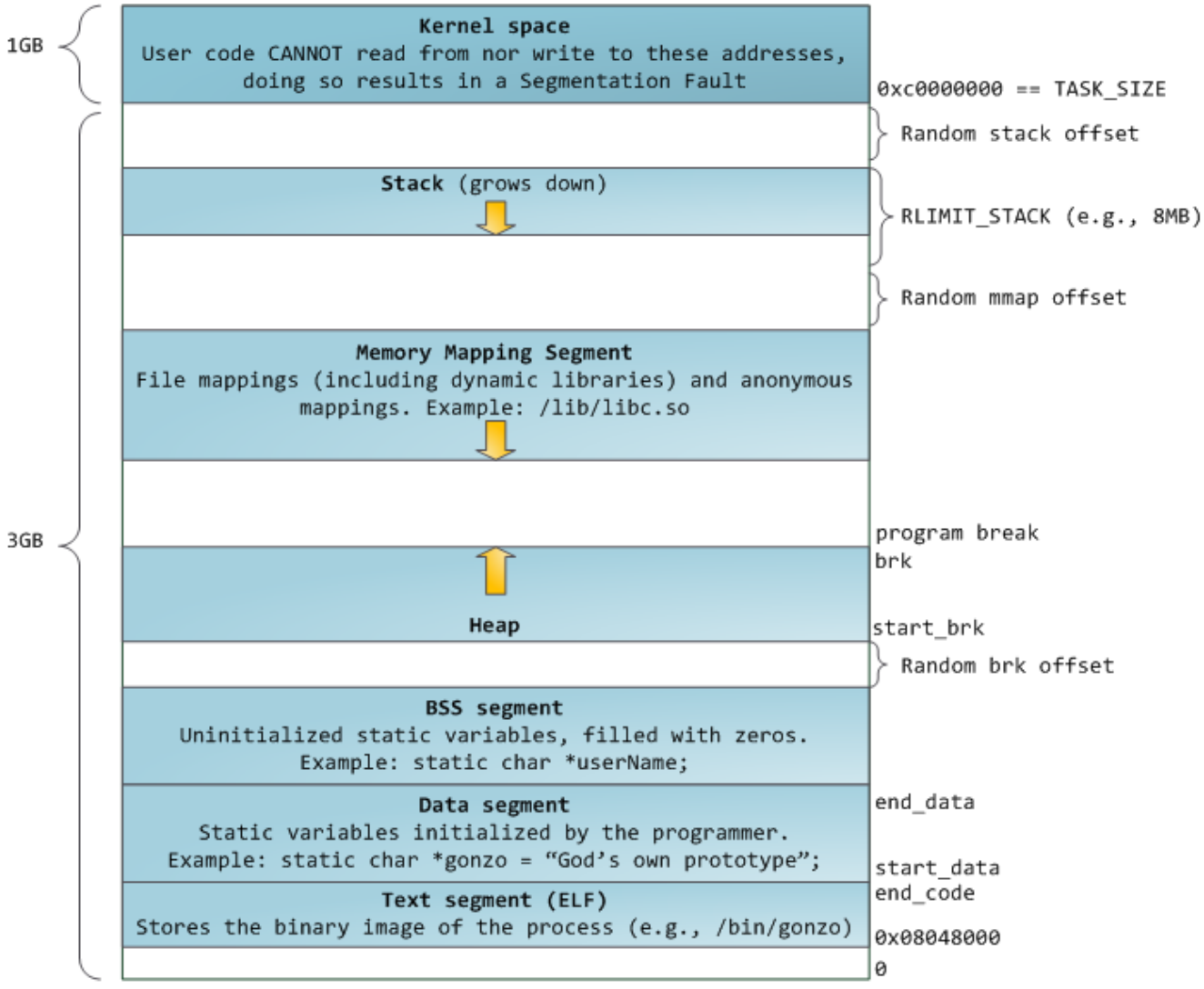


Figura 3.3: Organização da memória [Duarte, 2009]

Espaço de memória virtual (64 bit)

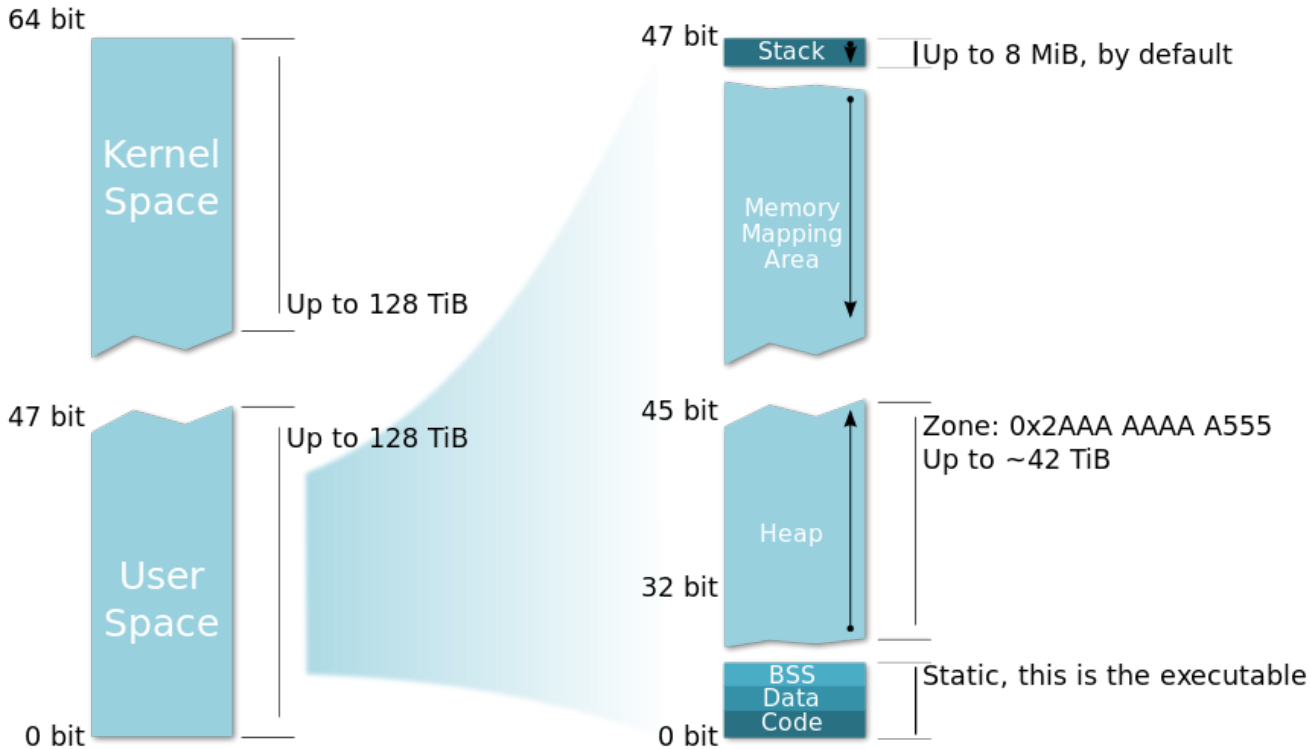


Figura 3.4: Organização da memória em 64 bit [2]

[2] Fonte: https://commons.wikimedia.org/wiki/File:Linux_Virtual_Memory_Layout_64bit.svg

Endereçamento em 64bit

- Somente os 48 bits menos significantes do processador são utilizados no endereçamento [3];
- Os bits 48 a 63 devem ser cópias do bit 47;
- O bit 47 contém a pilha de execução;
- Espaço total endereçável: 256 TB;
- Aumento de 64.000 vezes em relação aos processadores de 32 bit.

[3] Normalmente utilizadas para armazenamento da tabela de páginas


Algolist (2016).
Algorithms and data structures.
Disponível em: http://www.algolist.net/Data_structures/Binary_heap/Insertion Acessado em 11/04/2016.


Chagas, F. (2016).
Notas de aula do prof. fernando chagas.


Duarte, G. (2009).
Anatomy of a program in memory.
Disponível em: http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory/ Acessado em 11/04/2016.


FARINES, J. M. and MELO, R. (2000).
Sistemas de Tempo Real, volume 1.
IME-USP.


Favacho, A. (2009).
Notas de aula da Profa. Aletéia Favacho.


Guarezi, D. J. and Silva, E. B. (2010).
Processos em windows e unix.
Disponível em:
http://www.inf.ufsc.br/∼magro/PROCESSOS%20EM%20WINDOWS%2
Acessado em 28/01/2011.


Rezende, P. (2016).
Notas de aula do Prof. Pedro Rezende.
Disponível em: http://cic.unb.br/~rezende/tc.html Acessado em 14/03/2016.

OBRIGADO!!!
PERGUNTAS???