FAGAMMON
FACULDADE PRESBITERIANA GAMMON

Sistemas Operacionais

Gerência de Memória no Linux

Erasmo Evangelista de Oliveira
erasmo.oliveira@fagammon.edu.br

Lavras, 2021

Introdução

- O Linux é um sistema operacional com memória virtual paginada, isto quer dizer que podemos ter programas em execução cujo tamanho é maior que a memória física disponível para executá-los.
- O sistema operacional passa a ser responsável por manter na memória as partes dos programas efetivamente em uso, deixando o resto no disco rígido

Introdução

- No Linux, a gerência de memória é subdividida em dois componentes:
  - Gerência de Memória Física: alocação e liberação de memória física (páginas, grupos de páginas e pequenos blocos de memória)
  - Gerência de Memória virtual.
- O Linux é um s**istema operacional com memória virtual paginada**, isto quer dizer que podemos ter programas em execução cujo tamanho é maior que a memória física disponível para executá-los.
- O sistema operacional passa a ser responsável por **manter na memória as partes dos programas efetivamente em uso**, deixando o **resto no disco rígido.**

Introdução

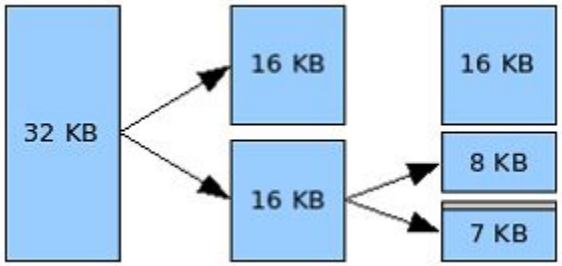
- Por exemplo, um programa de 16MB pode ser executado em uma máquina de 4MB de memória, com o sistema operacional selecionando os 4MB do programa que deverão ser mantidos na memória a cada instante, com as suas partes sendo copiadas do disco rígido para a memória e vice-versa, quando necessário.

Gerência de Memória Física

- Responsável pela alocação e liberação de todas as páginas físicas e capaz de disponibilizar intervalos de páginas fisicamente contíguas sob demanda
- Algoritmo buddy-heap (companheiro) para rastrear as páginas físicas disponíveis
  - Divide a memória em partições para tentar satisfazer uma requisição de memória da forma mais adequada possível.
    - Divisão da memória em metades para tentar proporcionar um best-fit.
    - regiões alocáveis possuem uma parceira adjacente e sempre que duas regiões parceiras são liberadas, estas se combinam para construir uma região maior.
    - Solicitações de pequenos blocos que não puderem ser satisfeitas por não existir uma região disponível, resultam na divisão de uma região maior em duas outras parceiras de tamanho igual,.
    - Repete-se o processo se necessário, até encontrar uma região do tamanho desejado

Gerência de Memória Física

- Para uma solicitação de 7 kilobytes de memória em um alocador que utiliza um sistema de parceiros de tamanho binário. Nesse caso, 1 kilobyte será perdido, pois o tamanho mínimo permitido capaz de suportar os 7 kilobytes é 8 kilobytes.



Buddy Algorithm

- O Linux gerencia a memória usando o algoritmo companheiro (buddy algorithm), com a adição de um vetor no qual o primeiro elemento é a cabeça de uma lista de blocos com tamanho de uma unidade.
- O segundo elemento é a cabeça de uma lista de blocos com tamanho de duas unidades.
- O próximo elemento aponta para blocos de quatro unidade e assim por diante. Dessa maneira qualquer bloco de potência de dois pode ser encontrado rapidamente.

Buddy Algorithm

- A vantagem do algoritmo do companheiro é que facilita a busca de bloco livre, se for implementada com um estrutura de árvore.
- No entanto, esse algoritmo gera uma considerável fragmentação interna, pois, se você deseja um bloco de 65 páginas, você tem de solicitar e obter um bloco de 128 páginas.

O Problema da Fragmentação

- A fragmentação é resolvida pelo Kernel com um processo de desfragmentação que junta espaço preenchidos de memória que são categoricamente semelhantes.
- Isto é feito com uma segunda alocação de memória que obtêm blocos, usando o algoritmo companheiro, e depois os retalha (unidades menores) para gerenciar unidades menores separadamente.

Gerência de Memória Virtual

- No Linux, o responsável pela manutenção do espaço de endereçamento visível para cada processo é o sistema de memória virtual.
- A criação das páginas de memória virtual sob demanda e a gerência do carregamento dessas páginas para o disco, ou o descarregamento de volta para o disco, é responsabilidade desse sistema.
- o gerente de memória virtual mantém duas perspectivas do espaço de endereçamento de um processo:
  - **visão lógica**: descreve as instruções recebidas pelo sistema de memória virtual referentes ao layout do espaço de endereçamento.
  - **visão física**: armazenada nas tabelas de páginas do hardware para o processo e é gerenciada por um conjunto de rotinas.

Gerência de Memória Virtual

- No Linux, o responsável pela manutenção do espaço de endereçamento visível para cada processo é o sistema de memória virtual.
- A criação das páginas de memória virtual sob demanda e a gerência do carregamento dessas páginas para o disco, ou o descarregamento de volta para o disco, é responsabilidade desse sistema.
- o gerente de memória virtual mantém duas perspectivas do espaço de endereçamento de um processo:
  - **visão lógica**: descreve as instruções recebidas pelo sistema de memória virtual referentes ao layout do espaço de endereçamento.
  - **visão física**: armazenada nas tabelas de páginas do hardware para o processo e é gerenciada por um conjunto de rotinas.

Processos e a Memória

- No Linux, processos que estão em execução têm prioridade na memória, quando termina um processo e havendo espaço na memória, ficam resíduos desse processo para uma futura volta desse processo ser mais rápida.
- Os dados não são apagados imediatamente após o encerramento da execução. Isso garante maior agilidade na execução dos processos.

Processos e a Memória

- Cada processo do Linux, em uma máquina de 32 bits, dispões de 3GB de espaço de endereçamento virtual para si próprio, com 1GB restante reservado para suas tabelas de páginas e outros dados do núcleo.
- O 1GB do núcleo não é visível quando o processo executa no modo usuário, mas torna-se acessível quando o processo faz uma chamada ao núcleo.

Processos e a Memória

- O espaço de endereçamento é gerado quando o processo é criado e sobrescrito em uma chamada ao sistema (exec).
- Chamadas de sistema:
  - **fork**: Criar um novo processo;
  - **wait**: Aguardar o término de seu filho;
  - **exec**: Executar outro programa;
  - **exit**: Terminar sua execução;

Memória de Swap

- Caso a memória RAM esteja lotada com processos que estão em execução, aí começa a utilização da memória SWAP (troca).
- Daí a grande importância que o Linux dispensa a este espaço, ao ponto de criar uma partição exclusiva para este fim.

Espaço de Endereçamento Virtual

- O espaço de endereçamento virtual é dividido em áreas ou regiões organizadas em páginas. Contíguas e homogêneas.
- Isso quer dizer que cada área consiste de uma série de páginas consecutivas com proteção e propriedades de paginação idênticas.

Espaço de Endereçamento Virtual

- O segmento de código e os arquivos mapeados são exemplos de áreas.
- Pode haver vazios no espaço de endereçamento virtual entre essas áreas.
- Qualquer referência à memória para um vazio resulta em uma falta de página fatal (Page fault).
- O tamanho da página é fixo.

Swapping

- a memória principal não é suficiente para manter todos os processos ativos, então os processos excedentes devem ser colocados no disco e voltar à execução na memória principal dinamicamente
- o **swapping** usada para algoritmos de escalonamento, baseando-se em prioridades. Se um processo de prioridade mais alta necessitar ser carregado, o gerenciador de memória poderá descarregar um outro com prioridade mais baixa para o disco, para que o de maior prioridade possa ser executado.
-

Swapping

- quando o escalonador de CPU executar um processo, ele chama o *dispatcher*, que verifica se o próximo processo na fila está na memória.
- Se o processo não estiver na memória e não houver região de memória livre, o dispatcher descarrega um processo que está na memória **(swap out)** e carrega o processo desejado em seu lugar **(swap in)**, recarregando, então, os registradores de forma usual e transferindo o controle para o processo selecionado.

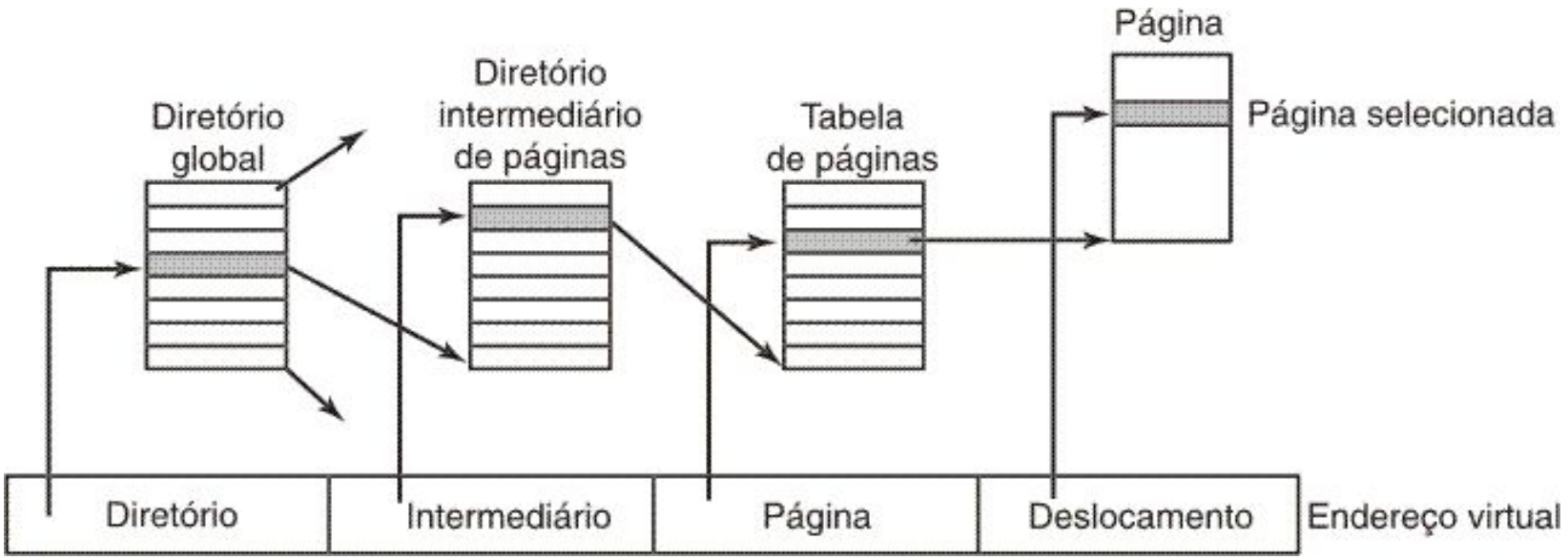
Paginação no Linux

- O Linux usa um esquema de paginação de três níveis que também é empregado de maneira modificada em várias arquitetura.
- Cada endereço virtual é quebrado em até quatro campos.
- O campo diretório é usado como índice do diretório global, sendo que existe um privado para cada processo.

Paginação no Linux

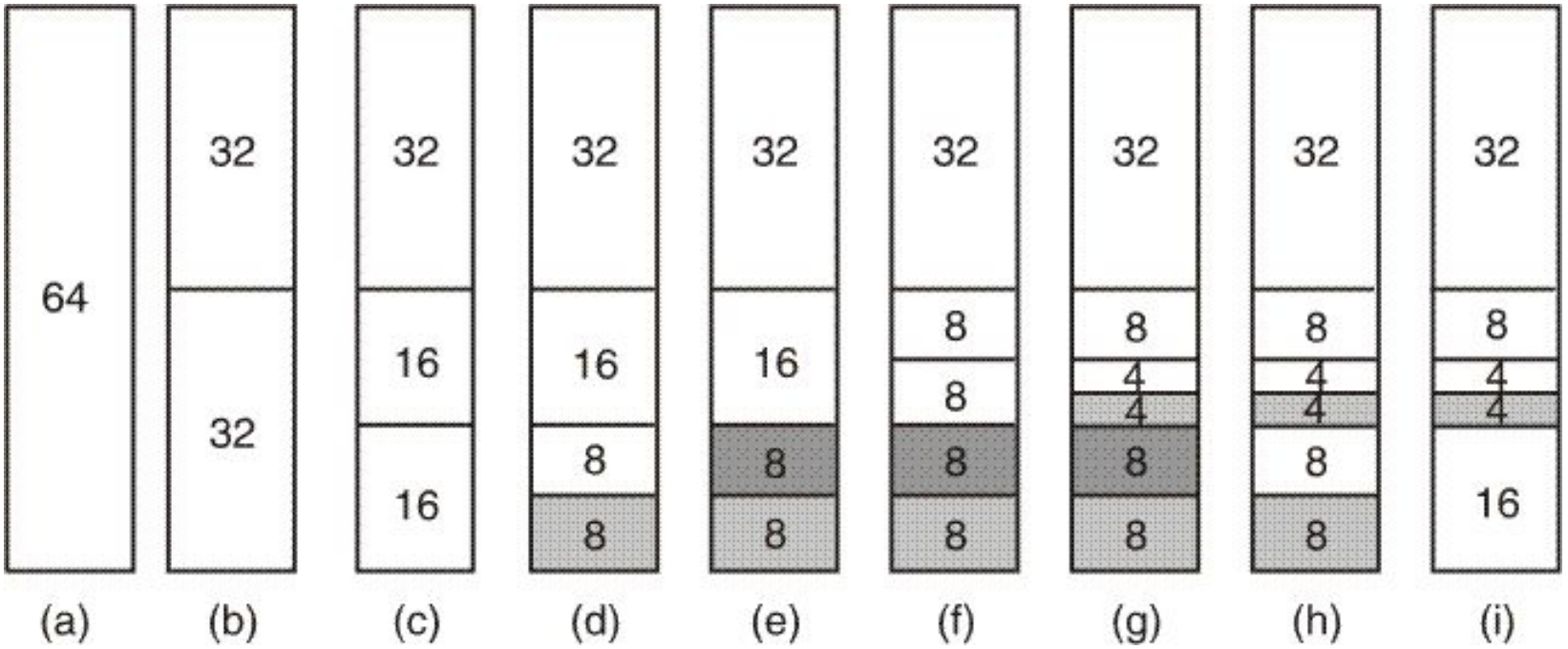
- Processo de paginação pode ser dividido em duas seções:
  - algoritmo de políticas, primeiramente, que decide que páginas são gravadas no disco e quando esse processo será feito, por meio de uma versão modificada do algoritmo de relógio, que emprega um relógio de passagens múltiplas.
  - o mecanismo de paginação, que suporta tanto partições e dispositivos dedicados, quanto arquivos, sendo que no último, o processo pode ser mais lento devido ao custo adicional provocado pelo sistema de arquivos.
    - O algoritmo utilizado para a gravação das páginas é o algoritmo next-fit, para tentar gravar páginas em carreiras contínuas de blocos de disco, visando um melhor desempenho.
-

Paginação no Linux



O Linux usa tabelas de páginas de três níveis

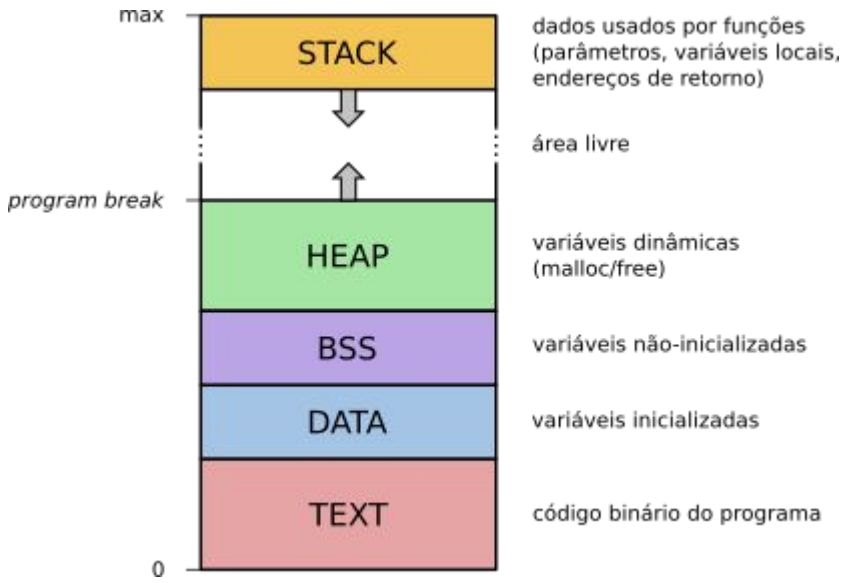
Paginação no Linux



Operação do algoritmo companheiro *(buddy)*

Memória de um Processo

- Como há dados de diversas naturezas, a memória de um processo é dividida em quatro segmentos bem definidos, de forma que cada região de memória possui características próprias.



Memória de um Processo

- O espaço de endereços de um processo em execução é dividido em várias áreas distintas. As mais importantes são:
  - Text: contém o código do programa e suas constantes. Esta área é alocada durante a chamada exec e permanece do mesmo tamanho durante toda a vida do processo.
  - Data e BSS: são as áreas onde o processo armazena suas variáveis globais e estáticas. Têm tamanho fixo durante a execução do processo.
  - Stack: contém a pilha de execução, onde são armazenadas os parâmetros, endereços de retorno e variáveis locais de funções. Pode variar de tamanho durante a execução do processo.
  - Heap: contém áreas de memória alocadas a pedido do processo, durante sua execução. Varia de tamanho durante a vida do processo.

Memória de um Processo

- **Código ou Text:** Neste segmento de memória estão as instruções que o processo usará e o processador executará. Seu tamanho e conteúdo são fixos, ou seja, não são alterados durante execução e são gerados durante o processo de compilação. Este segmento possui permissão para ser executado e lido, não podendo ser alterado.

- **Dados ou Data:** Segmento de memória que armazena as variáveis globais e estáticas de um processo, também possui um tamanho fixo visto que as variáveis desta região são definidas durante o processo de compilação. Este segmento pode ser lido e alterado, porém não pode ser executado como o segmento de código.

Memória de um Processo

- **Pilha ou Stack:** Neste segmento estão as regiões de memória locadas durante a execução do processo. Como esse segmento possui dados gerados durante a execução, seu tamanho varia durante esta execução e o segmento possui permissões de leitura e escrita, porém não possui permissão de execução.
- Os segmentos de pilha e heap crescem para uma região em comum, isso ocorre para garantir uma melhor utilização da memória disponível ao processo, porém os segmentos podem acabar se chocando, de forma que um deles acaba escrevendo dados no outro. Esta situação é conhecida como **Stack Overflow.**

Memória de um Processo

- **Heap:** Este segmento de memória armazena a pilha de execução do processo, ou seja, armazena informações referentes às chamadas de funções, aos valores de retorno destas, às variáveis locais de cada função e aos parâmetros das chamadas de funções. Como no segmento de pilha seu conteúdo pode ser lido e alterado, porém não pode ser executado e seu tamanho é alterado durante a execução do processo.

Comandos no Linux para Administração de Memória

- **free**
- **memstat**
- **vmstat**
- **dmesg**
- **pmap**
- **dmidecod**
- **htop**
- **top**

free

free: Mostra espaços livres e ocupados da memória RAM e SWAP

Parâmetros:
-b Visualiza os dados em bytes
-k Visualiza os dados em kilobytes (padrão)
-m Visualiza os dados em megabytes
-g Visualiza os dados em gigabytes

Referências

- SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G**. Fundamentos de Sistemas Operacionais**. 8ª Ed. Rio de Janeiro: LTC, 2010.

- TANENBAUM, Andrew S. **Sistemas Operacionais Modernos.** 3ª ed. Curitiba: Pearson. 2009.

Gravação da Aula

- Link da aula de 30/03/2020: https://bit.ly/2wR8qEh