



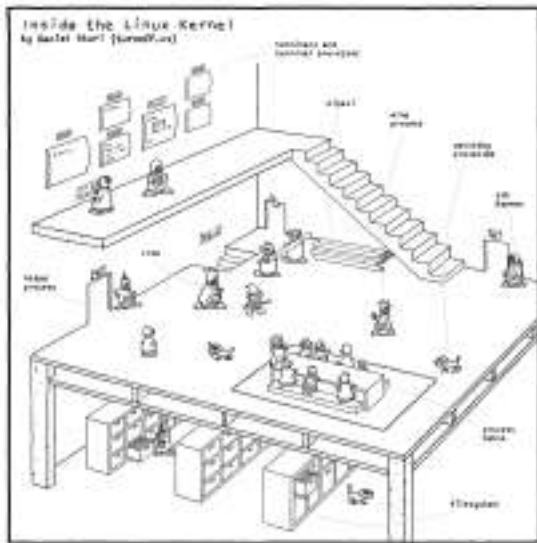
UNIVERSIDADE
VILA VELHA
ESPIRITO SANTO

Processos

Prof. Jean-Rémi Bourguet

Sistemas Operacionais

Inside the kernel



<https://turnoff.us/>

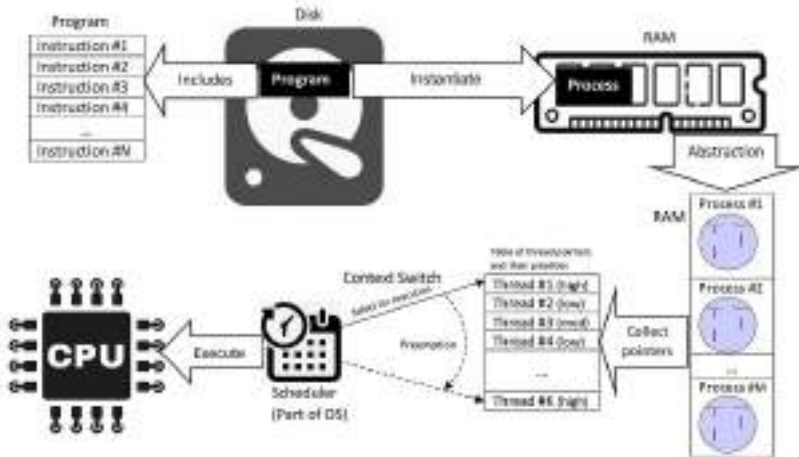
Inside the kernel

- ▶ O **conceito central** em qualquer sistema operacional é o de **processo**.



Inside the kernel

- **Instância em execução** de um **programa executável iniciado**.

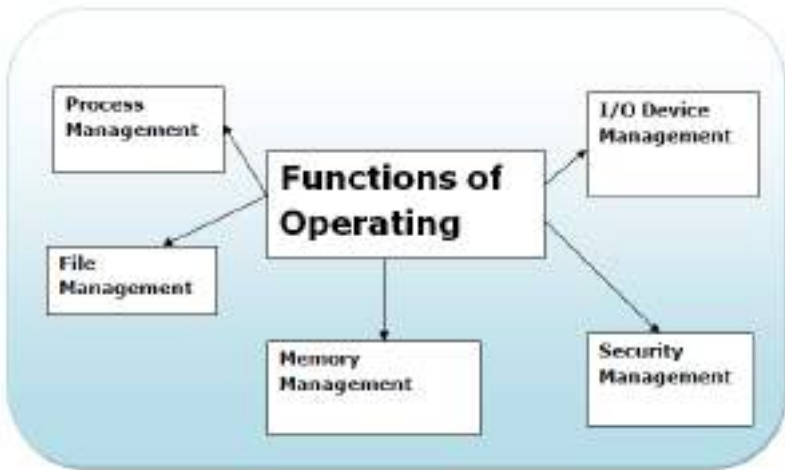


 <https://www.youtube.com/watch?v=C3dLwtwdEjM> (1983)

 <https://www.youtube.com/watch?v=AkFi90lZmXA> (2013)

Gerenciamentos de um sistema operacional

- As **funções de um OS** são definidas principalmente em **5 categorias**.



<https://conceptsall.com/what-are-the-functions-of-an-operating-system/>

Gerenciamentos de um sistema operacional

1. Gerenciamento de dispositivos I/O (Device Drivers)

- Aceita a entrada do **disp de entrada**, fornecer resultado aos de saída.



2. Sistema de gerenciamento de arquivos

- FS na memória secundária: criação e exclusão de diretório e arquivos.



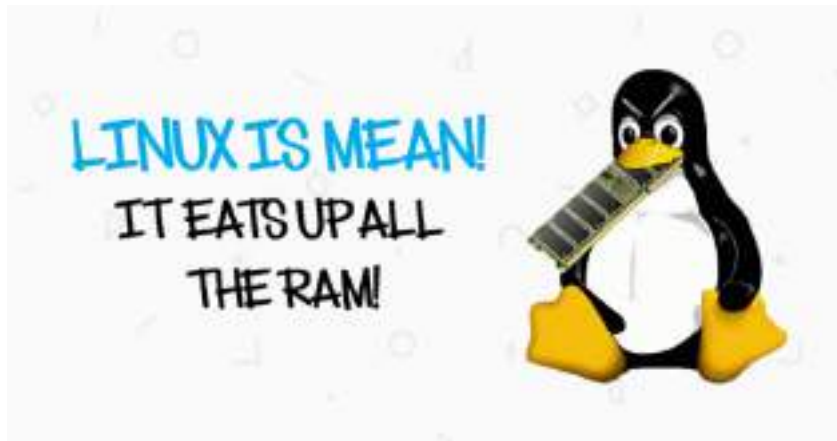
3. Gerenciamento de processos

- ▶ Usado para **gerenciar a CPU** definindo processos de programas
- ▶ **Criar, agendar e encerrar** o processo usado pela CPU.



4. Gerenciamento de memória

- ▶ Gerenciar as **alocações** e **desalocações** de endereços de memória.
- ▶ **Memória principal** é o acesso direto da CPU a um programa.



Gerenciamentos de um sistema operacional

5. Gerenciamento de segurança

- Antivírus e firewall para bloquear e permitir atividades de rede.



🏆 2. Gerenciamentos de um sistema operacional

Time sharing

- ▶ **CPU alterna** de um pra outro executando cada um em **milissegundos**.
- ▶ **Pseudoparalelismo: ilusão de paralelismo** contrasta com hardware.



Identificação de processos (PID)

- Processos **identificados** por um **número PID** (Process **ID**entifier).
- Pelo PID, o **sistema** consegue **controlar** e **interferir nos processos**.



🔺 RHA Chapter 8: Monitoramento e gerenciamento de processos

- **Procura** por todos os processos com nome e retorna o seu ID.

```
$ pgrep find  
11122
```



- **/proc** é o sistema de arquivos do Kernel do GNU/Linux.

Linux /proc

1 /proc

- 1 I mean everything in a file...
- 2 I use /proc file system to obtain information for running processes
- 3 Basic: it's a virtual file system
- 4 I know it doesn't contain real files, but I know it's

2 ls -l /proc

ls -l /proc is just a description, you can query this just a directory

ls -l /proc

drwxr-xr-x 5 root root 4 Feb 9 20
drwxr-xr-x 5 root root 4 Feb 9 20

3 Linux utilities that translate to commands

lsmod → /proc/modules
lsusb → /proc/bus/usb/lshw
lsblk → /proc/blockdev

4 <pid>

/proc/<pid>/cmdline - and the rest of process <pid>

/proc/<pid>/environ - environment variables to process <pid>

/proc/<pid>/fd - file descriptors used by pid

/proc/<pid>/stat - statistics of pid

/proc/<pid>/statm - process status

5 <pid>/task/<tid>

Information regarding thread of a specific process

/proc/<pid>/task/<tid>/comm - command, command the thread used to start, for thread with name all ways

/proc/<pid>/task/<tid>/children - children list of this thread, all task

6 What is it used for?

Get running & functionality to kernel

used to track system usability of kernel

Selection to kernel + build system LKM for programs

Diretório /proc

- ▶ **Digite ls** e veja a **quantidade de arquivos e diretórios** que ele possui.
- ▶ **Diretórios com números (pid)** com dados dos **processos em execução**.

```

1839 1485 19987 21915 23824 2563 5460 exccdomains self
1843 1478 1995 21921 23918 2564 542 fs xlaminfo
1844 1556 1996 2194 2395 26 55 filesystems settings
1846 1581 1997 21961 23953 2610 5487 fs stat
1850 1529 1998 21912 2396 2612 5610 interrupts swapi
1864 1598 2 2286 2397 2612 6 tomox dyn
1882 15993 2884 2229 2544 2614 6224 ioperfs sysrq-trigger
1887 15984 2881 22234 26 2616 625 irq sysvipc
1893 16819 2882 224 5 27 7817 kallsyms timer_list
1124 16411 2423 225 36 28 7818 kcore timer_stats
1125 1686 2825 226 32 29 792 kcp-users tty
1232 1739 2426 22688 326 6885 894 kmsg uptime
1342 1857 2432 22618 327 481 947 kpagecact version
1144 1876 2833 227 33 41 949 kpagecact version
1245 1888 2438 22729 34 4133 873 kpagecact version
1195 1931 2441 228 35 42 acpi kpagecact version
1288 1932 2458 229 3543 43 kpagecact version
12284 1835 2455 23262 3545 48 buddyinfo kpagecact version
1231 1936 2468 2328 3548 45 kpagecact version
1232 1938 2469 2331 3548 46 cgroups kpagecact version
1237 1942 2466 23441 3549 47 cgroups module
12758 1958 2468 23441 3551 481 cgroups mounts
12762 1966 2483 23452 3552 4928 cgroups attr
12843 1969 2227 23476 3554 5128 cgroups net
1281 1976 21586 23483 3555 516 devices pagetypeinfo

```


Diretório /proc

- ▶ **Entre em um diretório:** `$ cd /proc/$(pgrep firefox).`
- ▶ **Verifique as opções** usadas com **cat** (e.g. `cmdline`, `status`, `oom_score`).

an amazing directory: **/proc** Source: Exercks @BARK

Every process on Linux has a PID (process ID) like 42. In <code>/proc/42</code> , there's a lot of VERY useful information about process 42.	/proc/PID/cmdline command line arguments the process was started with	/proc/PID/exe symlink to the process's binary magic: works even if the binary has been deleted!
	/proc/PID/environ all of the process's environment variables	/proc/PID/status Is the program running or asleep? How much memory is it using? And much more!
/proc/PID/fd Directory with every file the process has open! Run <code>\$ ls -l /proc/42/fd</code> to see the list of files for process 42. These symlinks are also magic! If you can use them to recover deleted files!♥	/proc/PID/stack The kernel's current stack for the process. Useful if it's stuck in a system call	and more: look at man proc for more information!
	/proc/PID/maps List of process's memory maps: Shared libraries, heap, anonymous maps, etc.	

Vida dos processos

- ▶ O processo passa o essencial da sua vida **tentando acessar o CPU!**



Vida dos processos

- ▶ Conceitualmente, **cada processo** tem sua **própria CPU virtual**.
- ▶ Porém só existe **um só CPU** na máquina!



Vida dos processos

- ▶ O **processador não sabe** qual programa encontra-se em execução.
- ▶ **Concorrência** entre processos **alternando execução** de instruções.



Vida dos processos

- ▶ Todas as **informações do processo** interrompido são **salvas**.
- ▶ **Continuar a execução** no ponto onde estava quando foi interrompido.



Vida dos processos

Existe apenas um contador de programa físico para o CPU:

+ Processo é executado: **contador** do prog **lógico** é carregado no **físico**.

BORN TO FRAG



- Quando ele termina: o **contador** do prog **físico** é salvo no **lógico**.

- ▶ A **troca de um processo por outro** chama-se **mudança de contexto**.
- ▶ Conjunto de info de um processo chama-se **contexto do processo**.

CONTEXT SWITCHING



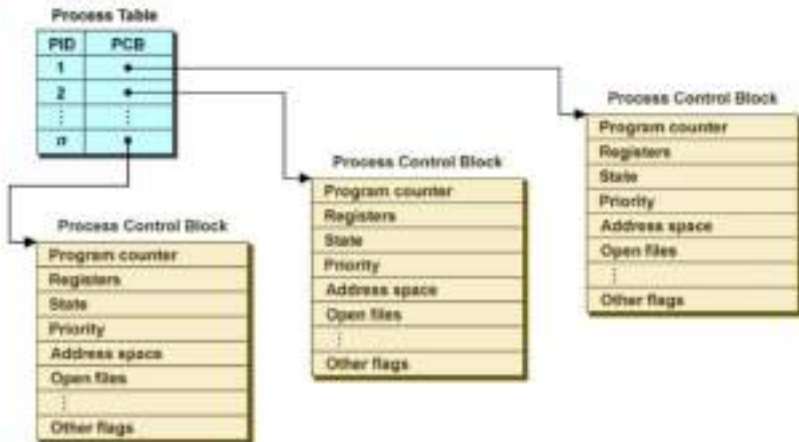
Bloco de Controle de Processo

- ▶ Os PCBs são **mantidos na memória principal** (área exclusiva).
- ▶ Tamanho controlado por **número máximo de processos suportados**.



Tabela de processos

- ▶ **Ponteiros para o PCB** de cada processo para **acessá-lo rapidamente**.
- ▶ **Índice da entrada** da **tabela de processos** do processo é o **PID**.

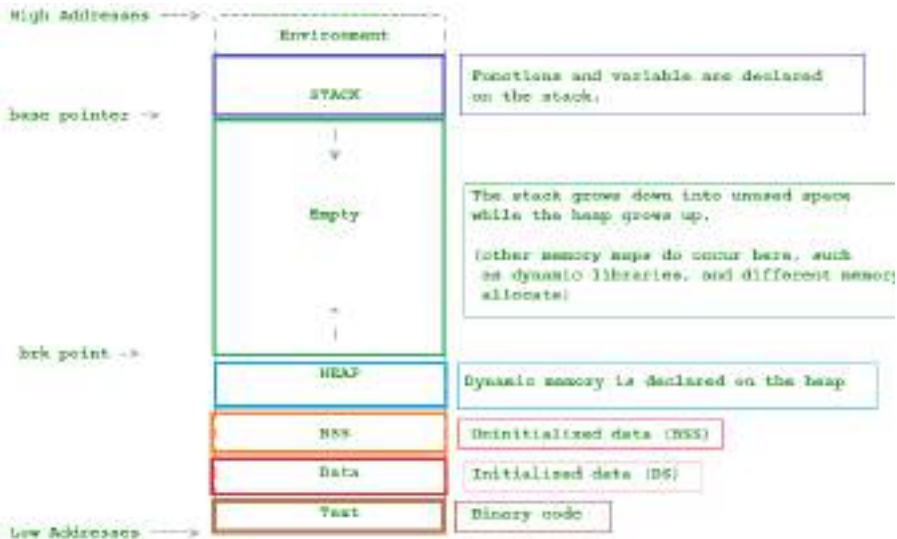


Contextos dos processos

- Cada processo possui seu **PCB** com todas as suas **informações**:
Contexto de hardware, de software e espaço de endereçamento.

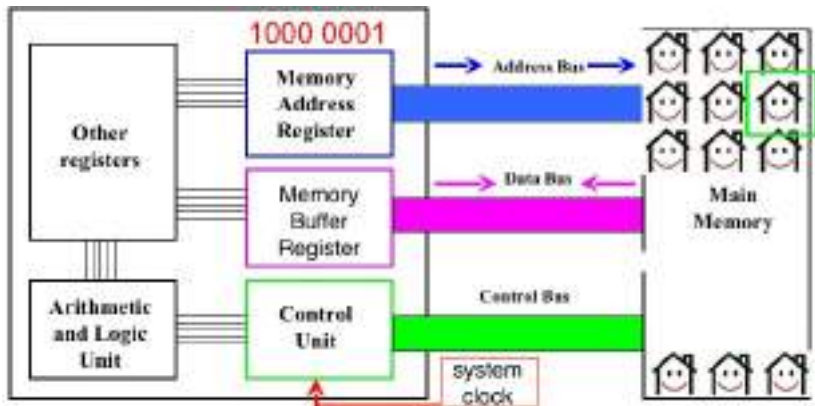


Contextos dos processos



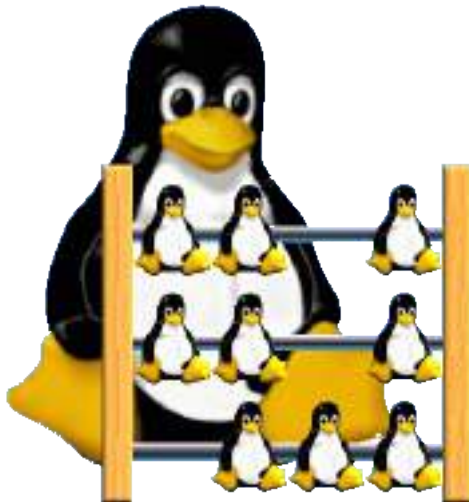
Contextos dos processos

- ▶ **Address Bus:**
Barramento de Endereço.
- ▶ **Data Bus:**
Barramento de Dados.
- ▶ **Instruction Bus:**
Barramento de Instruções.
- ▶ **Control Bus:**
Barramento de Controle.



2. Contexto de hardware

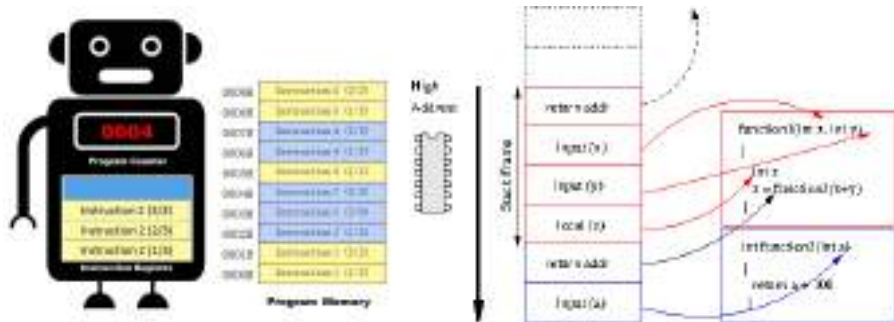
- ▶ Armazena o **conteúdo dos registradores e contadores** da **CPU**.
- ▶ São **carregados** antes que **processo retome sua execução** na CPU.



2. Contexto de hardware

Instruções de programas:

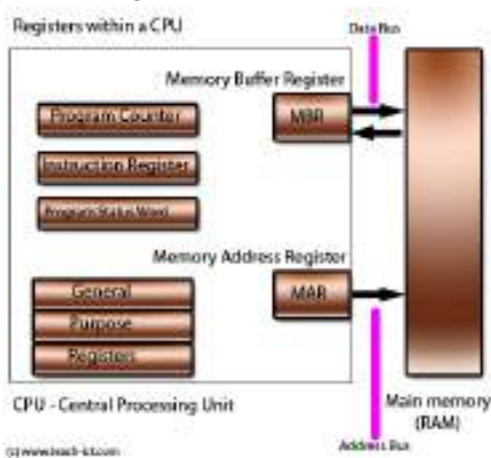
- ▶ **IR** (Instruction Register): instrução atual que está sendo executada.
- ▶ **PC** (Program Counter): endereço da próxima instrução a ser executada.
- ▶ **SP** (Stack Pointer): aponta para topo da pilha de memória do processo.



2. Contexto de hardware

Dados

- ▶ **MAR** (Memory Address Register): localização dados para leitura/gravação.
- ▶ **MBR** (Memory Buffer Register): dados sendo transferidos de e para RAM.



2. Contexto de hardware

Outros registradores:

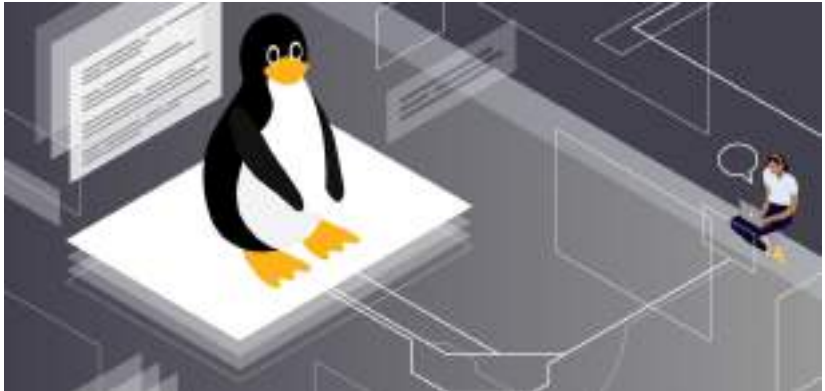
- ▶ **AC** (**AC**umulador): registrador das operações lógicas e aritméticas.
- ▶ **PSW** (**P**rogram **S**tatus **W**ord): estado do CPU e programa em execução.



3. Contexto de software

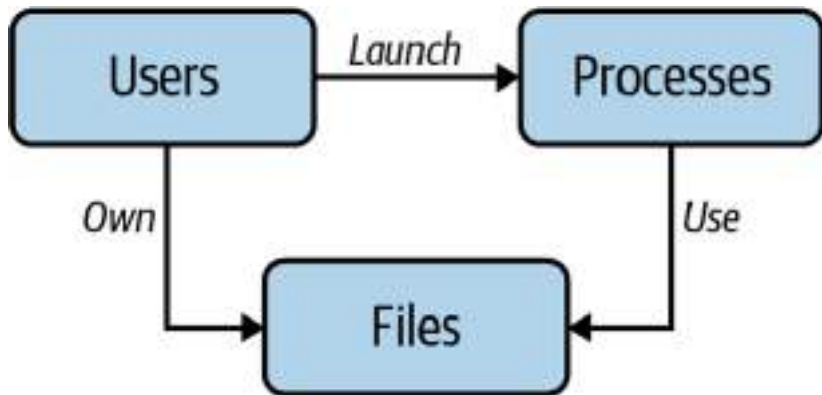
Características determinadas na criação e execução do processo:

- ▶ **Privilégios:** em relação ao sistema e aos outros processos.
- ▶ **Quotas:** limites de cada recurso que um processo pode alocar.



Privilégios

- ▶ Um processo com **privilégios elevados** acesso a recursos críticos.
- ▶ Um processo com **privilégios mais baixos** podem ter acesso restrito.



Quotas

- ▶ **Manter controle** tanto do **limite** quanto do **uso atual** de recursos.
- ▶ Garantir que um **processo não exceda os limites** definidos.



Exemplos de Limite (Quota)

- ▶ **Numero máximo** de **arquivos abertos** simultaneamente.
- ▶ **Tamanho máximo** da **memória principal e secundária** alocada.
- ▶ **Número máximo** de **operações** de **E/S pendentes**.
- ▶ **Tamanho máximo** do **buffer** para **operações de E/S**.



 https://linuxhint.com/set_max_user_processes_linux/

Uso Atual (Accounting Information)

- ▶ **User CPU Time:**

Quantidade de tempo CPU gasta na execução do processo.

- ▶ **System CPU Time:**

Execução das funções (kernel) conectadas ao seu processo.



4. Espaço de Endereçamento

- ▶ **Área de memória** do processo (cada um possui seu próprio espaço).
- ▶ **Protegido** pelo OS para que **não haja acesso** pelos **demaís processos**.



- Mostra¹ o **uso da memória** por parte dos **diferentes processos**.

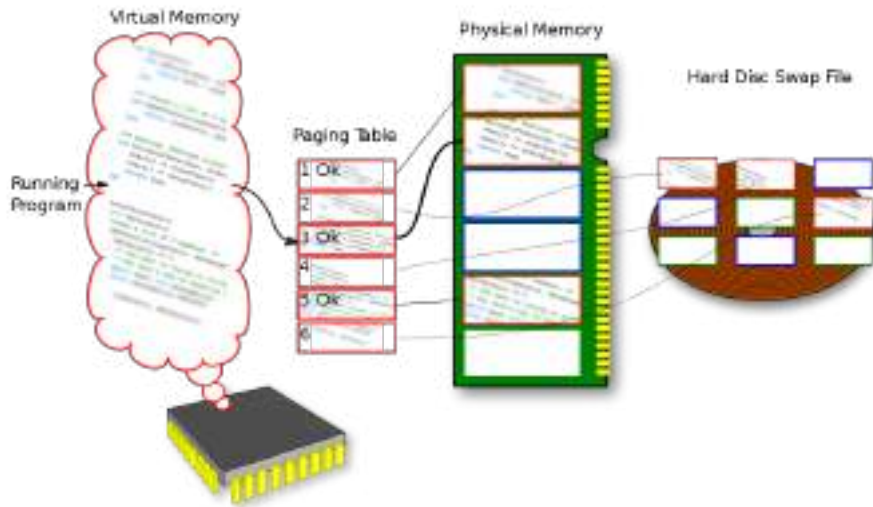
```
$ memstat | sort -nr | head -n 20
```

```
1082948k: PID 30047 (/usr/lib/firefox/firefox)
1061960k: PID 2795 (/usr/bin/nautilus)
991736k: PID 29117 (/usr/bin/evince)
990848k: PID 29885 (/usr/bin/evince)
791944k: PID 2685 (/usr/bin/compiz)
624308k: PID 2630 (/usr/lib/evolution/evolution-calendar)
547156k: PID 29065 (/usr/bin/texmaker)
542128k: PID 3653 (/usr/bin/zeitgeist-datahub)
```

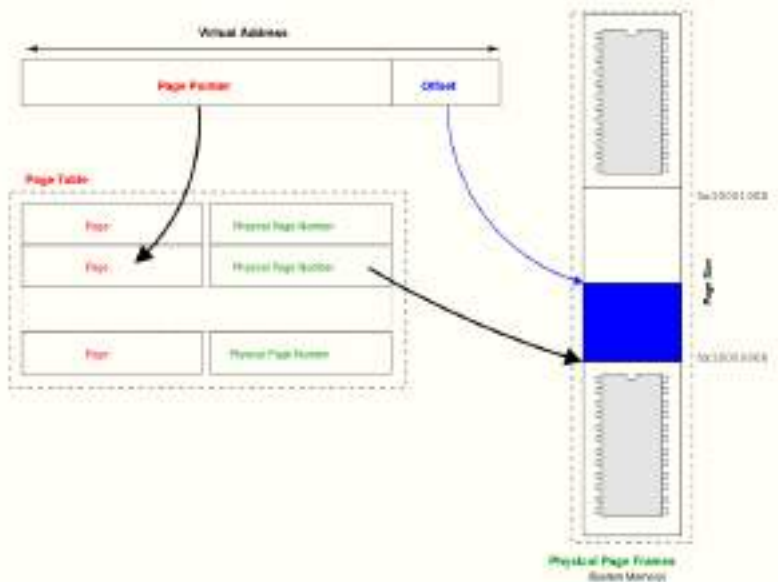
MEMSTAT

¹Instale com # apt-get install memstat

Páginas, Blocos e Palavras de memória



Páginas, Blocos e Palavras de memória

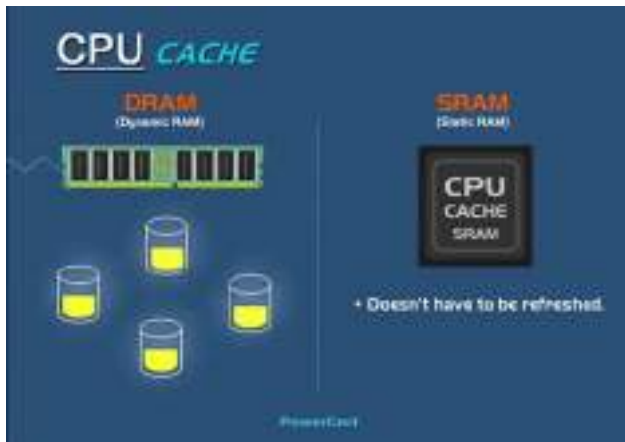
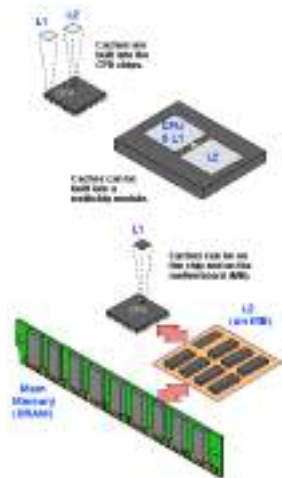


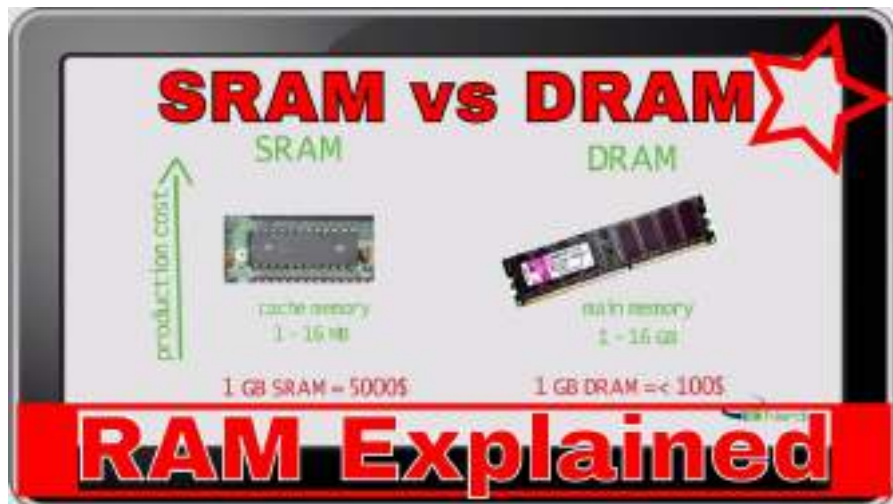
Páginas, Blocos e Palavras de memória

- ▶ **DRAM: D**ynamic **R**andom-**A**ccess **M**emory:
- ▶ **Carga elétrica vaza lentamente**, sem intervenção, dados são perdidos.
- ▶ Circuito de **atualização de memória**, reescreva periodicamente dados.



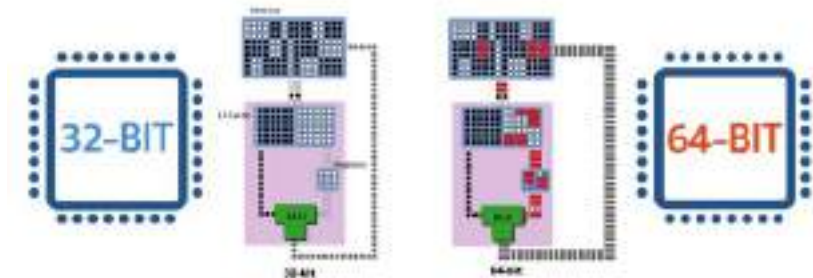
- **SRAM**: **S**tatic **R**andom-**A**ccess **M**emory:
Não exige que os dados sejam atualizados.





Páginas, Blocos e Palavras de memória

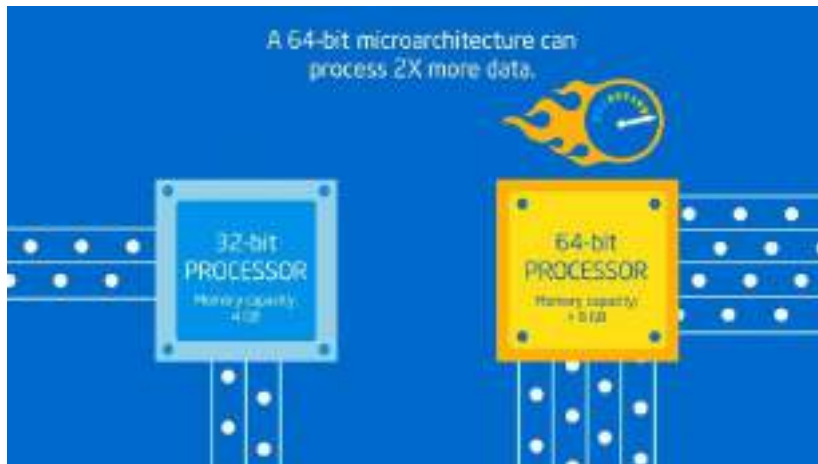
- ▶ As **memórias armazenam informações** em sequencia: as **palavras**.
- ▶ **Registradores** contém **uma palavra de dados** (32 ou 64 bits).



32 bit vs 64 bit

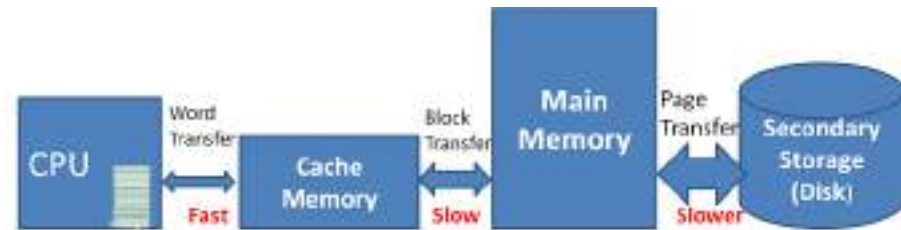
Páginas, Blocos e Palavras de memória

- ▶ CPU de 32 bits **não poderá** usar **mais de 4 GB** (2^{32}) de **RAM**.
- ▶ CPU de 64 bits **não poderá** usar **mais de 18.4 EB** (2^{64}) de **RAM**.



Páginas, Blocos e Palavras de memória

- ▶ Da **memória secundária**: transferência feita por **páginas/segmentos**.
- ▶ Da **memória primária** DRAM: transferência feita por **blocos**.
- ▶ Da **memória primária** SRAM: transferência feita por **palavras**.

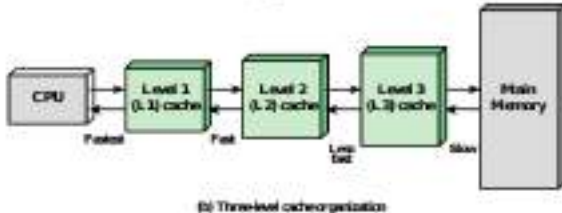
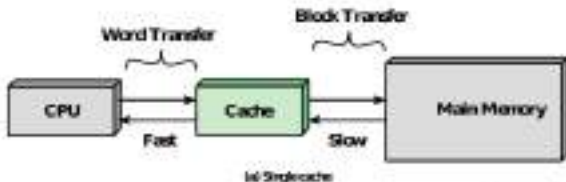


🏆 3. Gerência de memória

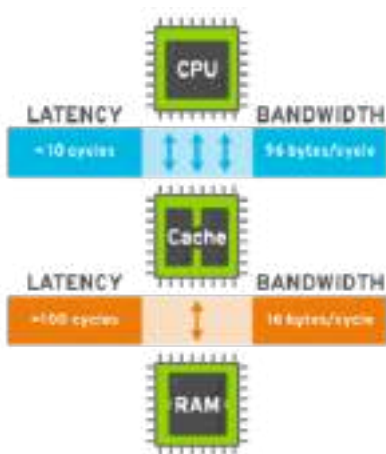
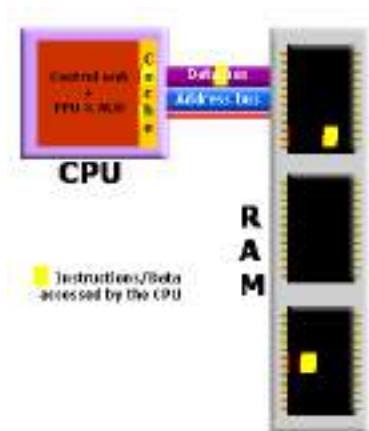
Buffer/Cache <-> DRAM

Quando o processador deseja ler uma palavra da memória:

- + Teste para determinar se a palavra está na **memória cache** (cacheada).
- Caso contrário **vai pegar da memória principal**.

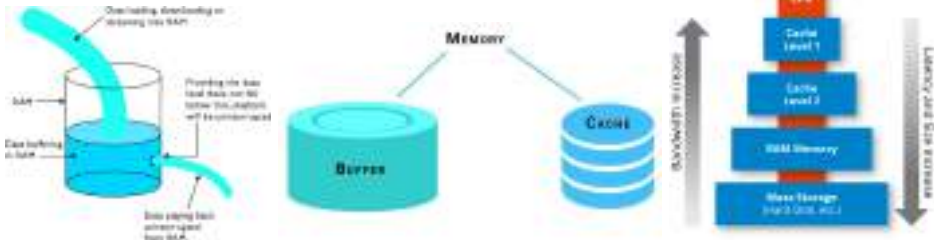


Buffer/Cache <-> DRAM



Buffer/Cache <-> DRAM

- **Caching** fixa a **velocidade de acesso dos dados** usados repetidamente.



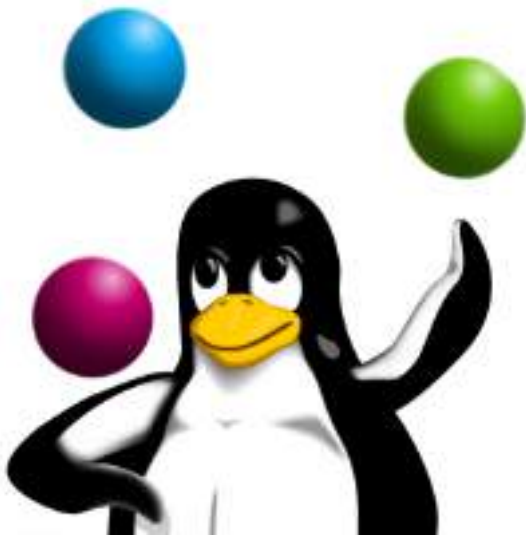
- **Buffering** agiliza a **velocidade do fluxo de dados** da **transmissão**.



<https://www.youtube.com/watch?v=UPefH41WQHM> (opt)

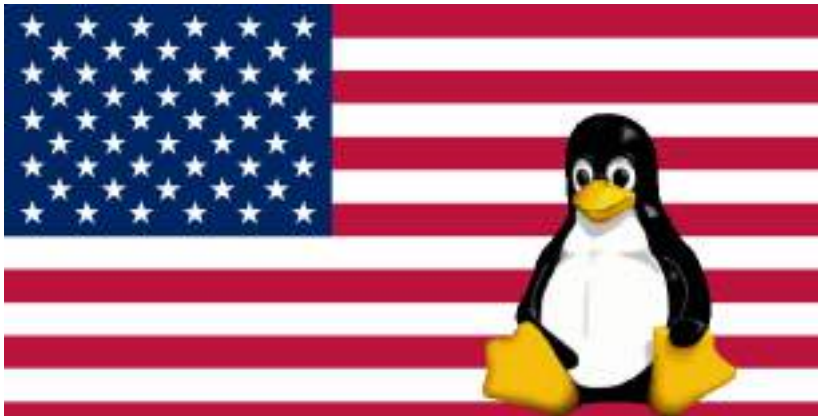
5. Estados de um Processo

- ▶ Na **vida do processo**: alocação de recursos/acesso ao CPU se alteram.



5. Estados de um Processo

- ▶ Processos passam por **vários estados** durante a sua execução.
- ▶ Isso em função de **eventos gerados pelo OS** ou pelo **próprio processo**.



5. Estados de um Processo

Um processo ativo pode encontrar-se em um de três diferentes estados:

- ▶ **Execução** (Running): Quando o processo está em execução pelo CPU.
- ▶ **Pronto** (Ready): Quando o processo aguarda para ser executado.
- ▶ **Espera** (Wait): Quando o processo aguarda um evento ou recurso.



5. Estados de um Processo

- ▶ O sistema operacional determina o **escalonamento de processos**.
- ▶ **Ordem** e **critério do acesso** ao CPU pelo processo em **estado pronto**.



🏆 4. Estados de processos

5. Estados de um Processo

- ▶ Processo pode **aguardar** o **termino da gravação** de um arquivo.
- ▶ Pode **esperar determinada hora** para iniciar a **execução de processo**.



Criação de Processos

- ▶ OS adiciona **novo PCB**, aloca espaço de **endereçamento na memória**.
- ▶ A partir da criação do PCB, OS já reconhece a **existência do processo**.



Criação de Processos

- ▶ Processo é dito no **estado de criação** quando OS já criou novo PCB.
- ▶ Porém, não pode colocá-lo na **lista de processos** de **estado pronto**.



Eliminação de Processos

- Eliminação de processo **desaloca os recursos** dele e **remove PCB**.

HANDLING NON-RESPONDING & FROZEN APPLICATIONS



Eliminação de Processos

- ▶ Tal processo não poderá mais ter **nenhum recurso alocado**.
- ▶ Porém o **OS** ainda **mantém informações** do **processo em memória**.



Descrição da média de carga

- ▶ A **média de carga** é uma medida fornecida pelo **kernel do Linux**.
- ▶ **Quantas solicitações** de recursos do sistema estão **pendentes**?

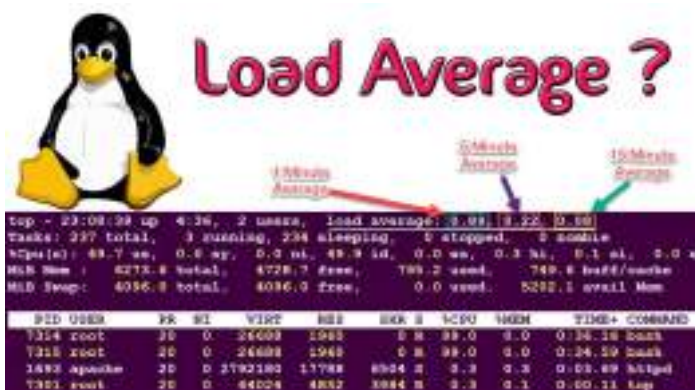


Descrição da média de carga

- ▶ A cada **cinco segundos**, quantos processos estão **aguardando**?
- ▶ Usa **média móvel exponencial** nos **últimos 1, 5 e 15 minutos**.

\$ **uptime**

15:29:03 up 14 min, 2 users, load average: 2.92, 4.48, 5.20



Descrição da média de carga

- ▶ **lscpu** pode ajudar a determinar **quantas CPUs** um sistema possui.
- ▶ Divide os valores de **média de carga** exibidos pelo **número de CPUs**.

\$ **lscpu**

```
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            4
```

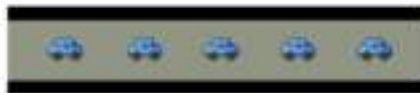
```
# From lscpu, the system has 4 logical CPUs, so divide by 4:
#                               load average: 2.92, 4.48, 5.20
#       divide by number of logical CPUs:      4      4      4
#                               _____
#                               per-CPU load average: 0.73   1.12   1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs,
# all CPUs were in use ~73% of the time.
# During the last 5 minutes, system was overloaded by ~12%.
# During the last 15 minutes, system was overloaded by ~30%.
```


Descrição da média de carga

- ▶ Um **valor abaixo de 1** indica o **uso satisfatório** dos recursos.
- ▶ Um **valor acima de 1** indica **saturação de recursos**.
- ▶ Uma **fila de CPU** inativa tem um **número de carga de 0**.



= load of 1.00



= load of 0.50



= load of 1.70

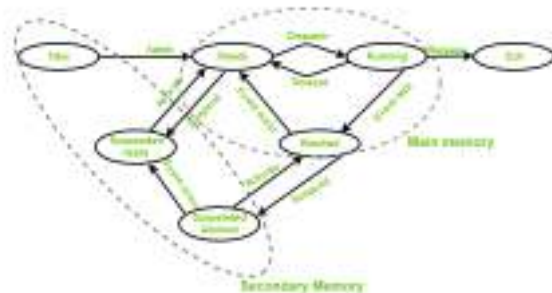
Swapping de Processos

- ▶ **Quantidade** de processos ativos pode ser **maior** da **capacidade** RAM.
- ▶ Causa **falta de recursos** e utiliza-se o artifício da **memória virtual**.



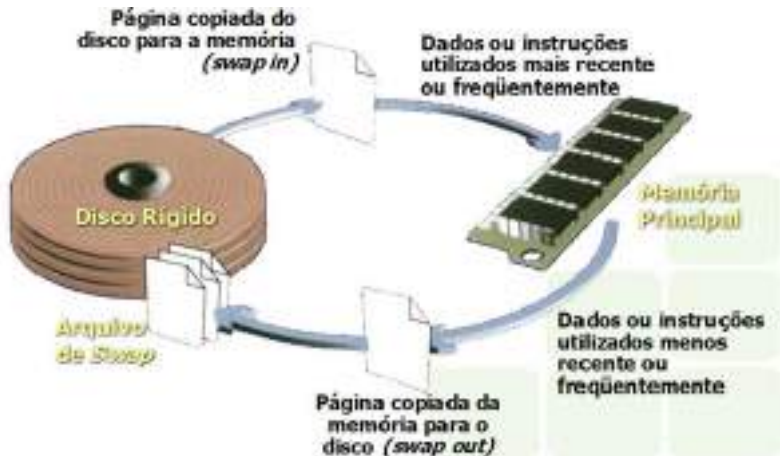
Swapping de Processos

- ▶ Se a **memória encher**: retirar processos da mem 1ª e **trazer de volta**.
- ▶ Alguns **processos esperando transferidos para HD** (área de swap).



Swapping de Processos

- ▶ Qualquer dado a ser processado deverá voltar **na RAM**.
- ▶ A **área swap** caracteriza-se como um **ponto de espera** e nada mais.



Swapping de Processos

- ▶ **Transferência** entre **RAM** (nanos $10^{-9}s$) e **HD** (ms $10^{-6}s$) é **lenta**.
- ▶ Fica **indesejável** o uso de swap, e deve justificar o **acréscimo da RAM**.



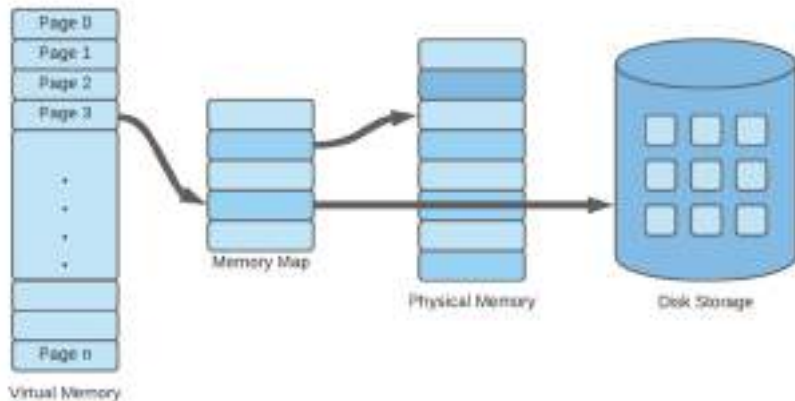
Paginação e Segmentação

- Métodos ou técnicas usadas para **alocação** de **memória não contígua**.



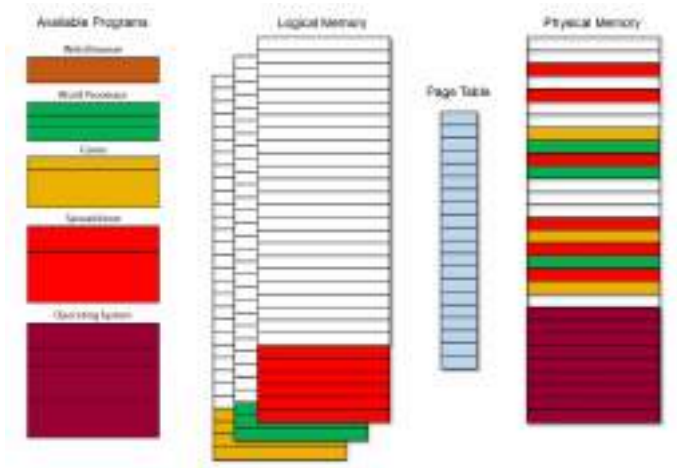
Paginação e Segmentação

- ▶ **Endereço lógico** é gerado pela CPU durante a **execução do programa**.
- ▶ O endereço lógico é um **endereço virtual**, pois **não existe fisicamente**.
- ▶ Referência para **acessar a localização** da memória física pela CPU.



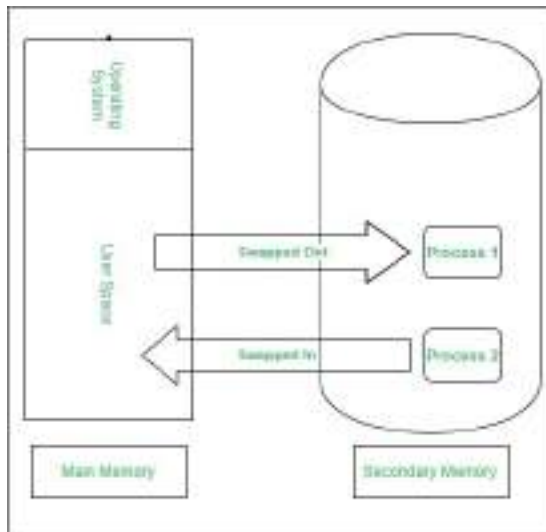
Paginação e Segmentação

- ▶ O seu **armazenamento não é necessariamente sequencial**.
- ▶ **Elimina a fragmentação externa** da memória.



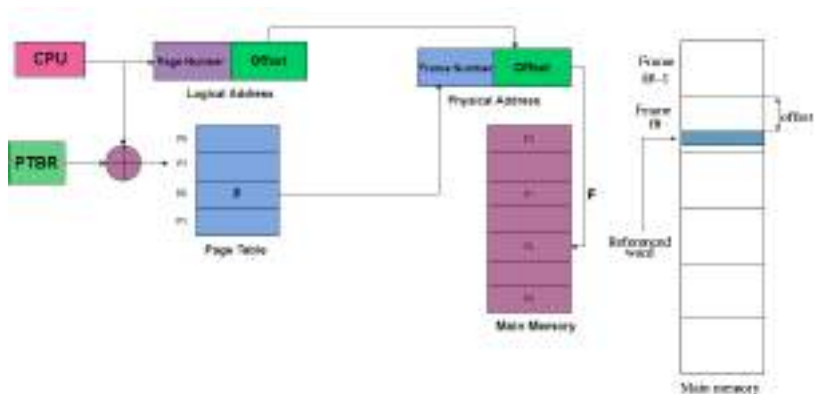
Paginação e Segmentação

- **Armazenamento e recuperação dados da memória 2^a para principal.**



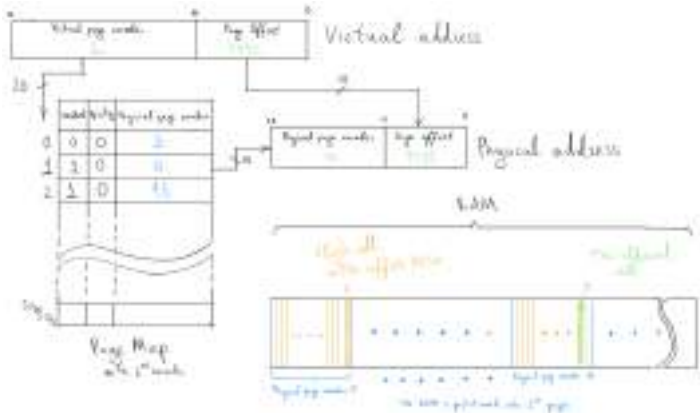
Paginação

- ▶ Paginação através de **consulta a tabelas**: Pagemaps/Page Tables.
- ▶ **Cada processo** tem um **PTBR**: Page-Table **B**ase **R**egister.



Endereço virtuais e físicos em 32 bits (20 + 12) divide-se em:

- **Número da página** codificada em **20 bits** ($2^{20} = 10100$ páginas).
 - Offset em **12 bits** ($2^{12} = 4096$ combinações para endereçar 1 Byte).
- Assim as páginas contêm um **tamanho fixo** de **4KB** de dados.



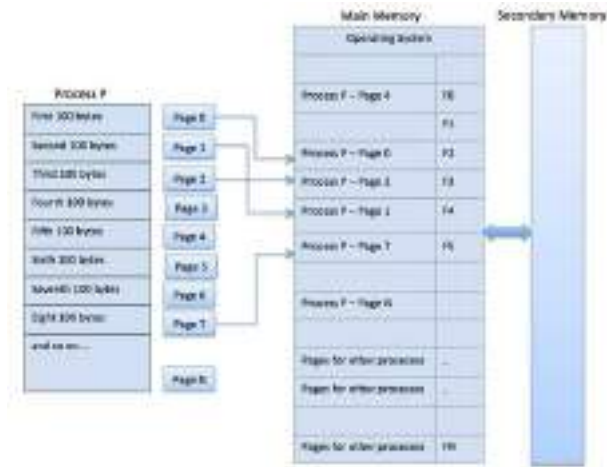
Paginação

- ▶ Pode **carregar as páginas** de código sob demanda durante a execução.
- ▶ Quando um processo é iniciado, apenas **parte inicial é carregada**.
- ▶ Outras páginas atingindo **novas regiões de código** são carregadas.



Paginação

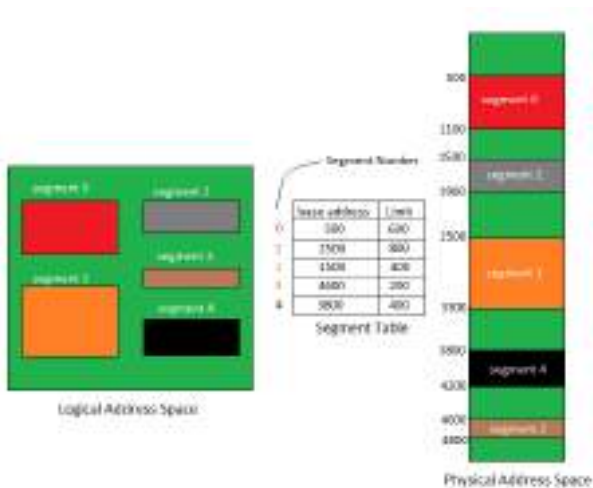
- Esquema de **particionamento de tamanho fixo**: as **páginas**.



5. Paginação

Segmentação

- **Partições** do processo de **tamanho variável**: os **segmentos**.



Segmentação

- Detalhes de cada segmento armazenados em **tabela de segmentação**.

