

Monitoramento em Ambientes Linux

Avaliação de Desempenho de Sistemas

Prof: Paulo Maciel

Instrutores: Carlos Melo, Paulo Pereira e Ronierison Maciel
{casm3,prps,rsm4,prmm}@cin.ufpe.br

Avaliação de Desempenho de Sistemas

- Por que Monitorar?
- O que Monitorar?
- **Como Monitorar?**
- **Onde começar?**

- In 2019, **100% of the world's supercomputers run on Linux.**
- Out of the **top 25** websites in the world, only 2 aren't using Linux.
- **96.3% of the world's top 1 million** servers run on Linux.
- **90% of all cloud infrastructure operates on Linux** and practically all **the best cloud hosts** use it.

Monitoramento LINUX

- Ferramentas Essenciais

- Shell
- /proc
- ps
- top
- uptime
- vmstat
- free
- **stress**
- **sysstat**
 - iostat
 - mpstat
 - pidstat
- **dstat**
- **tcpdump**
- **nmon**

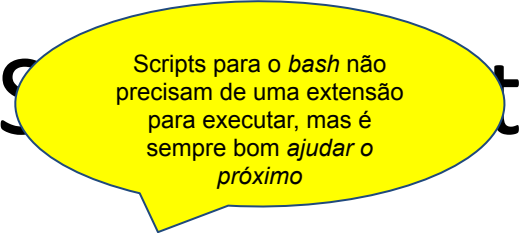
● requer instalação

O Shell

- É um programa que recebe comandos do teclado e os repassa ao sistema operacional.
- Uma interface de **linha de comando** para a biblioteca de comandos disponíveis no seu sistema operacional.

O Shell

- Linux
 - *bash*
- Windows
 - *bash*(WSL) e PowerShell
- Quando na GUI precisamos de outro programa para interagir com o shell, esses são os chamados **Emuladores de Terminal**
 - konsole
 - gnome-terminal

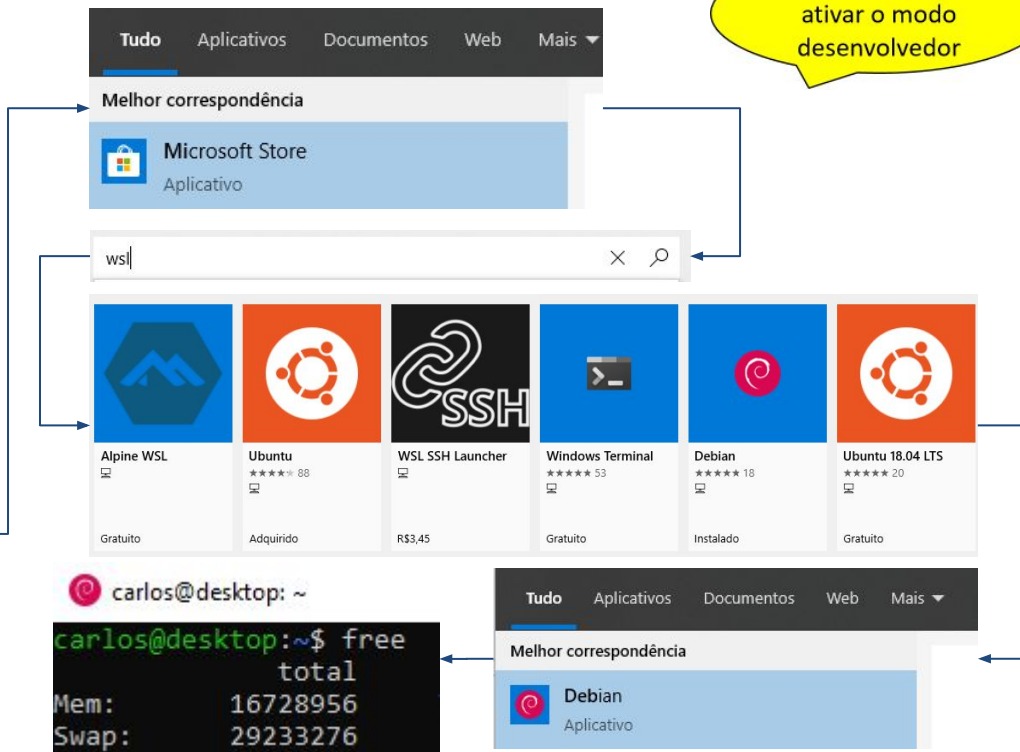
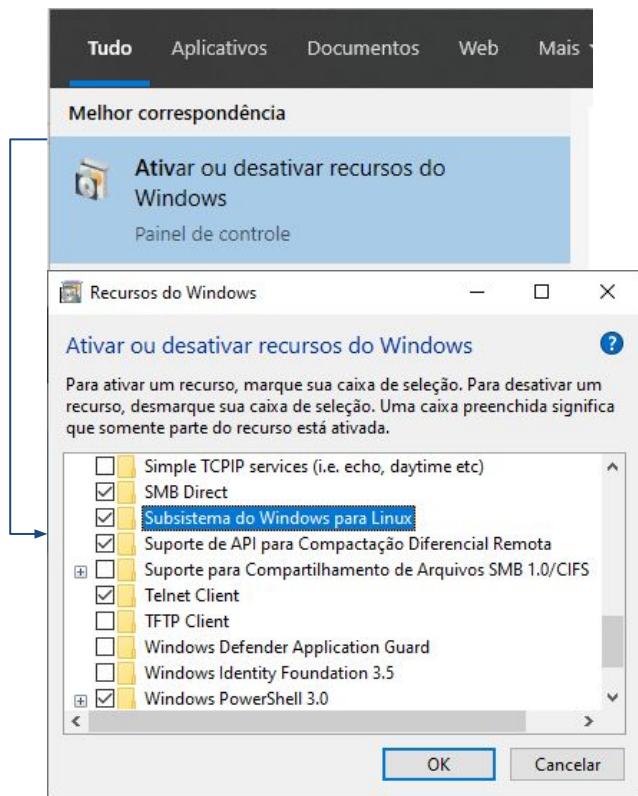


Scripts para o *bash* não
precisam de uma extensão
para executar, mas é
sempre bom *ajudar o*
próximo

- É um arquivo de texto (**.sh**, .ps1, ...);
- Automatiza tarefas específicas (tediosas e repetitivas);
- Executam um conjunto de comandos;
- Podem se valer de lógica simples (*for, while, if e else*).

Bash no Windows

Pode ser preciso
ativar o modo
desenvolvedor



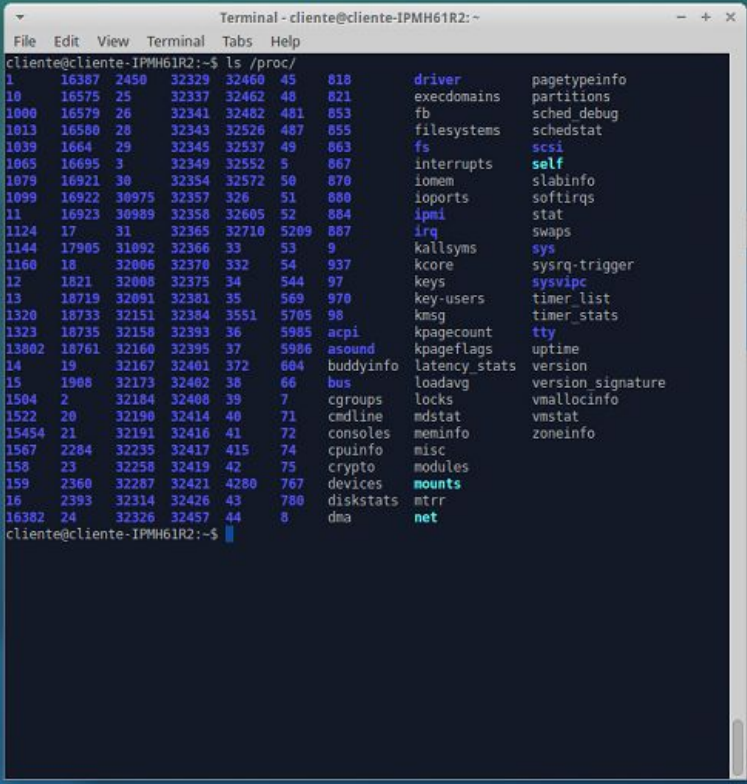
echo "Hello World!"

- Variáveis;
- Meu Primeiro Script;
- Permissão de Execução;
- sh script ou ./script?
- O #!/bin/bash
- Por que shell e não java, python, ruby, ...?

Utilizando o /proc

- /proc

- *pseudo*-sistema de arquivos, existente no GNU/Linux e em vários outros SOs baseados no Unix
- estruturado como uma hierarquia de diretórios e arquivos



```
Terminal - cliente@cliente-IPMH61R2: ~
File Edit View Terminal Tabs Help
cliente@cliente-IPMH61R2:~$ ls /proc/
1      16387 2450 32329 32460 45 818 driver pagetypeinfo
10     16575 25 32337 32462 48 821 execdomains partitions
1000   16579 26 32341 32482 481 853 fb sched_debug
1013   16580 28 32343 32526 487 855 filesystems schedstat
1039   1664 29 32345 32537 49 863 fs scsi
1065   16695 3 32349 32552 5 867 interrupts self
1079   16921 30 32354 32572 50 870 iomem slabinfo
1099   16922 30975 32357 326 51 880 ioports softirqs
11     16923 30989 32358 32605 52 884 ipmi stat
1124   17 31 32365 32710 5209 887 irq swaps
1144   17905 31092 32366 33 53 9 kallsyms sys
1160   18 32006 32370 332 54 937 kcore sysrq-trigger
12     1821 32008 32375 34 544 97 keys sysvipc
13     18719 32091 32381 35 569 970 key-users timer_list
1320   18733 32151 32384 3551 5705 98 kmsq timer_stats
1323   18735 32158 32393 36 5985 acpi kpagecount tty
13802  18761 32160 32395 37 5986 asound kpageflags uptime
14     19 32167 32401 372 604 buddyinfo latency_stats version
15     1908 32173 32402 38 66 bus loadavg version_signature
1504   2 32184 32408 39 7 cgroups locks vmallocinfo
1522   20 32190 32414 40 71 cmdline mdstat vmstat
15454  21 32191 32416 41 72 consoles meminfo zoneinfo
1567   2284 32235 32417 415 74 cpuinfo misc
158    23 32258 32419 42 75 crypto modules
159    2360 32287 32421 4280 767 devices mounts
16     2393 32314 32426 43 780 diskstats mtrr
16382  24 32326 32457 44 8 dma net
cliente@cliente-IPMH61R2:~$
```

Utilizando o /proc

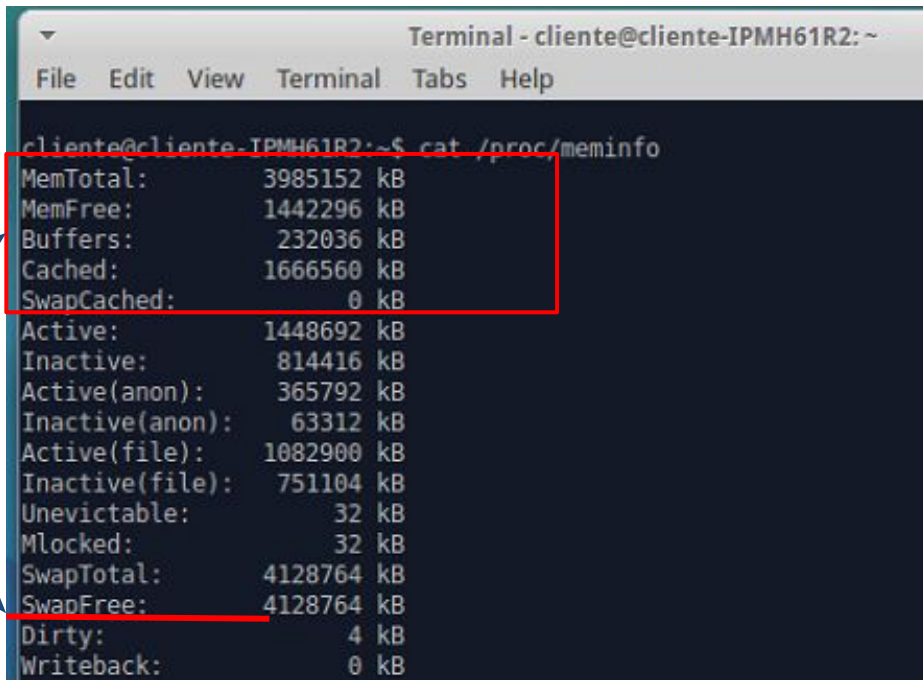
- /proc

- Interface para estruturas de dados internas do kernel (núcleo do sistema)
 - Acessar dados sobre processos e outros recursos do SO
 - Alterar parâmetros do kernel em tempo de execução
- Vários contadores de desempenho disponíveis
 - /proc/**stat**
 - /proc/**meminfo**
 - /proc/**vmstat**
 - /proc/**diskstats**
 - /proc/**net**/...
 - /proc/**<pid>**/...

/proc/meminfo

- /proc/meminfo

Informações
bastante úteis para
avaliar questões de
desempenho



A terminal window titled "Terminal - cliente@cliente-IPMH61R2: ~" showing the command `cat /proc/meminfo` and its output. A red rectangle highlights the first five lines of the output, and a red horizontal line underlines the `SwapFree` line. Two blue arrows originate from the text "Informações bastante úteis para avaliar questões de desempenho" and point to the `MemTotal` and `SwapFree` lines.

```
cliente@cliente-IPMH61R2:~$ cat /proc/meminfo
MemTotal:      3985152 kB
MemFree:       1442296 kB
Buffers:       232036 kB
Cached:        1666560 kB
SwapCached:      0 kB
Active:        1448692 kB
Inactive:      814416 kB
Active(anon):   365792 kB
Inactive(anon):  63312 kB
Active(file):  1082900 kB
Inactive(file): 751104 kB
Unevictable:    32 kB
Mlocked:        32 kB
SwapTotal:     4128764 kB
SwapFree:      4128764 kB
Dirty:          4 kB
Writeback:      0 kB
```

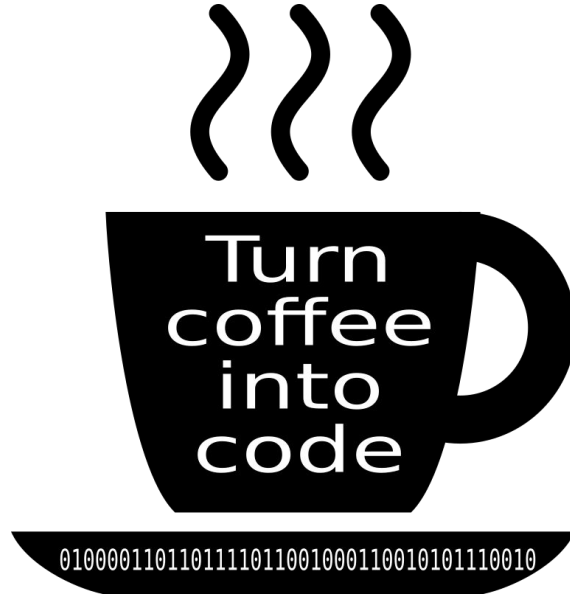
Exemplos /proc/meminfo

- Nós podemos monitorar em tempo real
- Ou salvamos em um arquivo (log) para visualizar depois

```
watch -n 5 cat /proc/meminfo
```

```
./monitora-memoria.sh
```

```
#!/bin/bash
tempo=0;
echo "MemFree Buffers Cached SwapFree">log-memoria.txt
while [ $tempo -lt 300 ]
do
    mfree=`cat /proc/meminfo | awk '/MemFree/{print $2}'`
    buff=`cat /proc/meminfo | awk '/Buffers/{print $2}'`
    cach=`cat /proc/meminfo | awk '/^Cached/{print $2}'`
    swapfree=`cat /proc/meminfo | awk '/SwapFree/{print $2}'`
    echo "$mfree $buff $cach $swapfree" >> log-memoria.txt
    sleep 5
    tempo=$((tempo + 5));
done
```



CODING TIME!!!

/proc/<pid>/status

- Em muitas situações, é essencial medir o uso de recursos para um processo em específico

VmSize: toda a memória virtual usada pelo processo.

VmHWM: teto atingido pelo RSS

Resident set size: memória física (RAM) usada pelo processo.

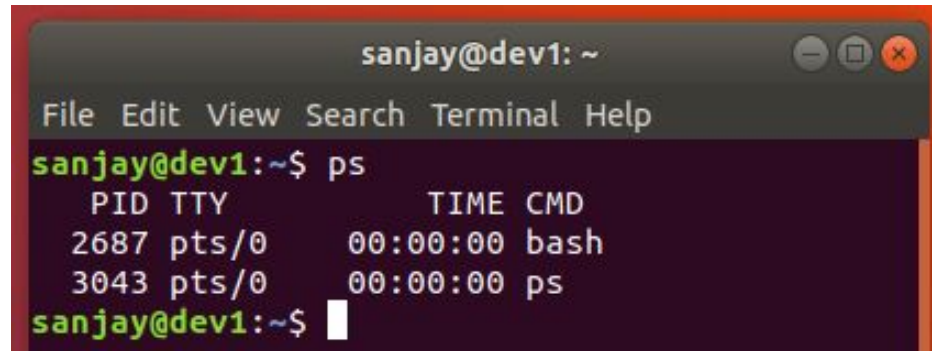
```
root@cliente-IPMH61R2:/home/cliente# cat /proc/18953/status
Name:      stress
State:     R (running)
Tgid:      18953
Ngid:      0
Pid:       18953
PPid:      18952
TracerPid: 0
Uid:       1000    1000    1000    1000
Gid:       1000    1000    1000    1000
FDSize:    64
Groups:    4 24 27 30 46 108 124 125 1000
VmPeak:    17680 kB
VmSize:    17680 kB
VmLck:     0 kB
VmPin:     0 kB
VmHWM:     10352 kB
VmRSS:     10308 kB
VmData:    10428 kB
VmStk:     136 kB
VmExe:     20 kB
VmLib:     2956 kB
VmPTE:     56 kB
VmSwap:    0 kB
Threads:   1
SigQ:      0/30965
SigPnd:    0000000000000000
chldPid:   0000000000000000
```

ps command (Process Status)

- O comando ps (status do processo) é um dos comandos mais usados no Linux. Geralmente é usado para obter informações mais detalhadas sobre um processo específico ou todos os processos.
- Por exemplo, é usado para saber se um processo específico está em execução ou não, quem está executando qual processo no sistema, qual processo está usando memória ou CPU mais alta, quanto tempo um processo está em execução etc.

ps command (Process Status)

- Sem nenhuma argumento, o comando **ps** mostra apenas o processo em execução na conta do usuário conectado no terminal atual.
- Você pode se perguntar por que o comando **ps** está mostrando dois processos enquanto ainda não executamos nenhum processo neste terminal.
 - O primeiro processo mostra o processo em que este terminal é aberto.
 - O segundo processo mostra o último comando executado neste terminal.

A terminal window titled 'sanjay@dev1: ~' with standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The prompt is 'sanjay@dev1:~\$'. The command 'ps' has been entered, and the output is displayed in a table format with columns for PID, TTY, TIME, and CMD. Two processes are listed: 'bash' (PID 2687) and 'ps' (PID 3043). The prompt 'sanjay@dev1:~\$' is shown again with a cursor.

```
sanjay@dev1: ~
File Edit View Search Terminal Help
sanjay@dev1:~$ ps
  PID TTY          TIME CMD
 2687 pts/0        00:00:00 bash
 3043 pts/0        00:00:00 ps
sanjay@dev1:~$
```


ps command (Process Status)

O comando **ps** aceita opções em três estilos.

- Estilo BSD UNIX: - Nesse estilo, as opções são fornecidas sem nenhum hífen inicial (como "aux").
- Estilo AT&T Unix: - Nesse estilo, as opções são fornecidas com um hífen principal (como "-aux").
- Estilo GNU Linux: - Nesse estilo, as opções são fornecidas com hífen iniciais duplos (como "--sort").

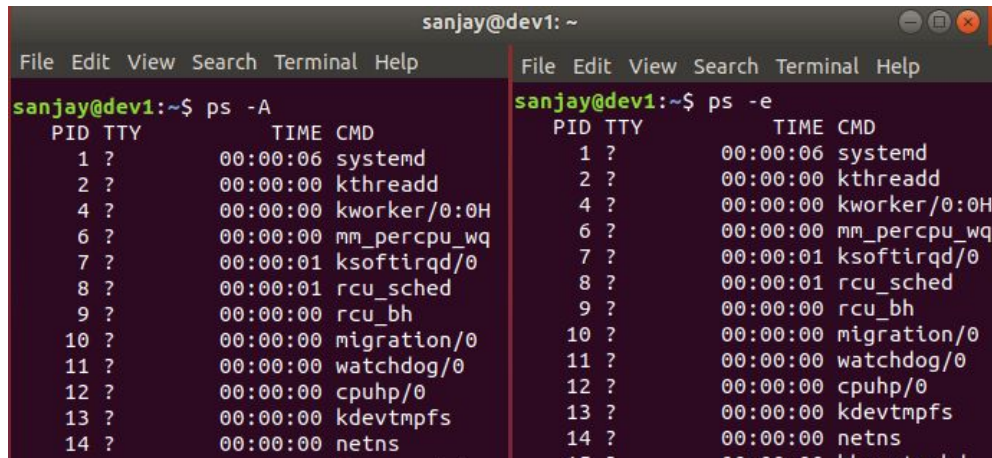
Embora o comando **ps** aceite opções no estilo de mixagem, você sempre deve usar apenas um estilo para especificar as opções.

ps command (Process Status)

- Para imprimir todos os processos em execução no sistema, use qualquer um dos seguintes comandos.

— \$ ps -A

— \$ ps -e



The image shows two side-by-side terminal windows. The left window displays the output of the command 'ps -A', and the right window displays the output of 'ps -e'. Both commands list system processes with columns for PID, TTY, TIME, and CMD.

PID	TTY	TIME	CMD
1	?	00:00:06	systemd
2	?	00:00:00	kthreadd
4	?	00:00:00	kworker/0:0H
6	?	00:00:00	mm_percpu_wq
7	?	00:00:01	ksoftirqd/0
8	?	00:00:01	rcu_sched
9	?	00:00:00	rcu_bh
10	?	00:00:00	migration/0
11	?	00:00:00	watchdog/0
12	?	00:00:00	cpuhp/0
13	?	00:00:00	kdevtmpfs
14	?	00:00:00	netns

PID	TTY	TIME	CMD
1	?	00:00:06	systemd
2	?	00:00:00	kthreadd
4	?	00:00:00	kworker/0:0H
6	?	00:00:00	mm_percpu_wq
7	?	00:00:01	ksoftirqd/0
8	?	00:00:01	rcu_sched
9	?	00:00:00	rcu_bh
10	?	00:00:00	migration/0
11	?	00:00:00	watchdog/0
12	?	00:00:00	cpuhp/0
13	?	00:00:00	kdevtmpfs
14	?	00:00:00	netns

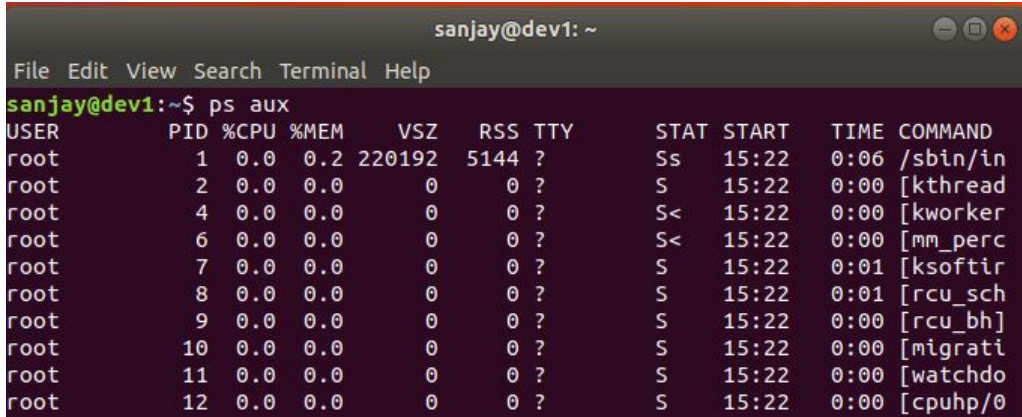
ps command (Process Status)

As opções A e e fornecem uma visão geral resumida dos processos em execução. Para imprimir a visão geral detalhada, use as opções f (formato completo) e F (formato extra extra) com essas opções.

```
sanjay@dev1: ~  
File Edit View Search Terminal Help  
sanjay@dev1:~$ ps -ef  
UID      PID    PPID  C  STIME TTY      TIME   CMD  
root         1        0  0  15:22 ?        00:00:06 /sbin/init splash  
root         2        0  0  15:22 ?        00:00:00 [kthreadd]  
root         4        2  0  15:22 ?        00:00:00 [kworker/0:0H]  
root         6        2  0  15:22 ?        00:00:00 [mm_percpu_wq]  
root         7        2  0  15:22 ?        00:00:01 [ksoftirqd/0]  
root         8        2  0  15:22 ?        00:00:01 [rcu_sched]  
  
sanjay@dev1: ~  
File Edit View Search Terminal Help  
sanjay@dev1:~$ ps -eF  
UID      PID    PPID  C   SZ  RSS  PSR  STIME TTY      TIME   CMD  
root         1        0  0 55048 5204   0  15:22 ?        00:00:06 /sbin/in  
root         2        0  0    0    0   0  15:22 ?        00:00:00 [kthread  
root         4        2  0    0    0   0  15:22 ?        00:00:00 [kworker  
root         6        2  0    0    0   0  15:22 ?        00:00:00 [mm_perc  
root         7        2  0    0    0   0  15:22 ?        00:00:01 [ksoftir  
root         8        2  0    0    0   0  15:22 ?        00:00:01 [rcu_sch  
root         0        2  0    0    0   0  15:22 ?        00:00:00 [rcu_bh]
```

ps command (Process Status)

Para visualizar a mesma saída no estilo BSD Unix, use as opções "aux".



```
sanjay@dev1: ~  
File Edit View Search Terminal Help  
sanjay@dev1:~$ ps aux  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
root         1  0.0  0.2 220192  5144 ?        Ss   15:22   0:06 /sbin/in  
root         2  0.0  0.0      0     0 ?        S    15:22   0:00 [kthread  
root         4  0.0  0.0      0     0 ?        S<   15:22   0:00 [kworker  
root         6  0.0  0.0      0     0 ?        S<   15:22   0:00 [mm_perc  
root         7  0.0  0.0      0     0 ?        S    15:22   0:01 [ksoftir  
root         8  0.0  0.0      0     0 ?        S    15:22   0:01 [rcu_sch  
root         9  0.0  0.0      0     0 ?        S    15:22   0:00 [rcu_bh]  
root        10  0.0  0.0      0     0 ?        S    15:22   0:00 [migrati  
root        11  0.0  0.0      0     0 ?        S    15:22   0:00 [watchdo  
root        12  0.0  0.0      0     0 ?        S    15:22   0:00 [cpuhp/0
```

ps command (Process Status)

O comando "ps aux" é o comando usado com mais frequência pelos administradores do Linux. Antes de passarmos para o próximo exemplo, vamos entender as opções usadas neste comando em detalhes.

- a: - Esta opção imprime os processos em execução de todos os usuários.
- u: - Esta opção mostra a coluna do usuário ou proprietário na saída.
- x: - Esta opção imprime os processos que não foram executados no terminal.

Coletivamente, as opções "aux" imprimem todo o processo em execução no sistema, independentemente de onde foram executadas.

Column	Description
USER	The user account under which this process is running
PID	Process ID of this process
%CPU	CPU time used by this process (in percentage).
%MEM	Physical memory used by this process (in percentage).
VSZ	Virtual memory used by this process (in bytes).
RSS	Resident Set Size, the non-swappable physical memory used by this process (in KiB)
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
STAT	Process state. <i>Explained in next table.</i>
START	Starting time and date of this process
TIME	Total CPU time used by this process
COMMAND	The command with all its arguments which started this process

D	uninterruptible sleep (usually IO)
R	running or runnable (on run queue)
S	interruptible sleep (waiting for an event to complete)
T	stopped by job control signal
t	stopped by debugger during the tracing
w	paging (not valid since the 2.6.xx kernel)
x	dead (should never be seen)
Z	defunct ("zombie") process, terminated but not reaped by its parent
<	high-priority (not nice to other users)
N	low-priority (nice to other users)
L	has pages locked into memory (for real-time and custom IO)
s	is a session leader
l	is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+	is in the foreground process group

ps command (Process Status)

- CPU é expresso como a porcentagem de tempo gasto em execução durante toda a vida útil de um processo.
- O RSS não conta algumas partes de um processo, incluindo as tabelas de páginas, pilha do kernel, struct thread_info e struct task_struct.
- VSZ é o tamanho virtual do processo (código + dados + pilha).
- Os processos marcados como <defunct> são processos mortos (os chamados "zumbis") que permanecem porque seus pais não os destruíram corretamente.

Se o comprimento do nome de usuário for maior que o comprimento da coluna de exibição, o nome de usuário será truncado.

awk command

- O awk é um utilitário que permite que um programador escreva programas pequenos, porém eficazes, na forma de instruções que definem padrões de texto a serem pesquisados em cada linha de um documento e a ação a ser tomada quando uma correspondência for encontrada dentro de um linha.
- O awk é usado principalmente para digitalização e processamento de padrões. Ele pesquisa um ou mais arquivos para ver se eles contêm linhas que correspondem aos padrões especificados e, em seguida, executa as ações associadas.
 - awk é abreviado dos nomes dos desenvolvedores - Aho, Weinberger e Kernighan.
 - `$ awk '/manager/{print $1}' employee.txt`



CODING TIME!!!

Monitoramento LINUX

- Comando **TOP**

- Fornece uma visão em tempo real do sistema em execução

- Sintaxe: top [opções]

- **-d** atraso Especifica o atraso em segundos entre as atualizações de tela. O padrão é 3 segundos.
 - **-i** ignora processos ociosos.
 - **-n** num Exibe num interações e depois termina.
 - **-b** Roda em modo de batch. Útil para mandar a saída de top para outros programas ou um arquivo.

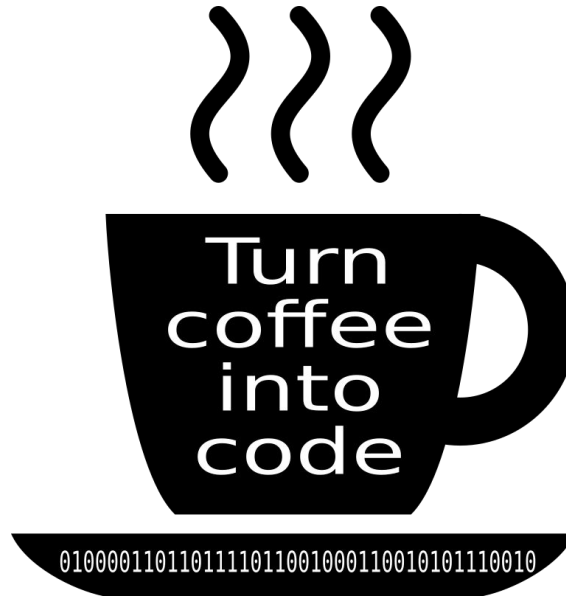
Monitoramento LINUX

Comando **top** – opções interativas

- h** Gera um tela de ajuda
- k** Termina um processo (será pedido seu PID)
- q** Sai do programa

Monitoramento LINUX

- **PID**: O identificador de cada processo
- **USER**: Usuário
- **PR**: Prioridade da Tarefa
- **NI**: Valor Nice da tarefa
- **VIRT**: Memória virtual usada
- **RES**: Memória física usada
- **SHR**: Memória compartilhada usada
- **S**: Estado da tarefa (**s** = sleeping, **R** = running, **T** = stopped, **Z** = zombie, etc.)
- **%CPU**: % de tempo de CPU
- **%MEM**: % de memória física
- **TIME+**: Tempo total de atividade da tarefa desde que ela foi iniciada
- **COMMAD**: Nome do processo

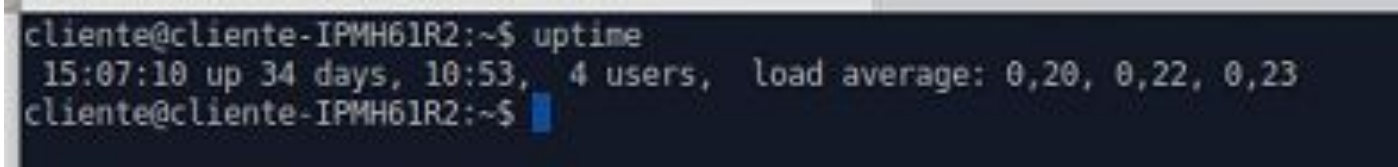


CODING TIME!!!

Monitoramento LINUX

- Comando **uptime**

–Mostra o tempo atual, há quanto tempo o **sistema** está **rodando**, quantos **usuários** estão **logados** atualmente e as médias de carga do sistema nos últimos 1, 5 e 15 minutos.



```
cliente@cliente-IPMH61R2:~$ uptime
15:07:10 up 34 days, 10:53,  4 users,  load average: 0,20, 0,22, 0,23
cliente@cliente-IPMH61R2:~$
```

Monitoramento LINUX

- Comando **vmstat**

- Este comando reporta informações sobre processos, memória, paginação, blocos de I/O, traps e atividades de CPU.
 - **Vmstat** [opções]
- **-S M** usa a unidade MB em vez do padrão KB
- **-a** Mostra memória ativa e inativa
- **-d** Mostra estatísticas de discos
- **-p** Partição Mostra informações de R/W na partição especificada
- **-s** Mostra estatísticas em formato de tabela

Monitoramento LINUX

Vmstat – campos

Procs

r: Nº de processos esperando para rodar

b: Nº de processos em dormência ininterrupta

Memory

Swpd: memória virtual usada

Free: memória livre

Buff: memória usada como buffer

Cache: memória usada como cache

Swap

si: memória trocada a partir do disco

so: memória trocada para o disco

Monitoramento LINUX

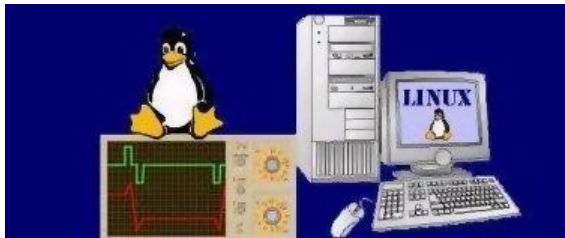
- **Vmstat** – campos

- io
 - **bi**: Blocos recebidos de um dispositivos de bloco (blocos/s)
 - **bo**: Blocos enviados a um dispositivo de bloco (blocos/s)
- System
 - **in**: nº de interrupções por segundo, incluindo clock
 - **cs**: nº de mudanças de contexto por segundo
- Cpu
 - **us**: Tempo gasto rodando código que não é kernel
 - **sy**: Tempo gasto rodando código do kernel
 - **id**: Tempo gasto em ociosidade
 - **wa**: Tempo gasto esperando por I/O

Monitoramento LINUX

- Comando **free**
 - Exibe a quantidade de memória livre e usada no sistema
- Sintaxe: free [opções]
 - b** Mostra o uso da memória em bytes
 - k** uso da memória em KB
 - m** em MB
 - t** Exibe uma linha que mostra os totais
 - s** n Operação contínua em intervalos de n segundos

Utilizando o Sysstat



- O **sysstat** é um pacote de utilitários para coleta de dados de desempenho
 - **iostat**: Disco e I/O em geral
 - **mpstat**: Processador e memória
 - **pidstat**: Monitoramento por processo

Monitoramento LINUX

- Comando **iostat**
 - Mostra informações sobre o uso da **CPU** e várias estatísticas sobre E/S do sistema.
- Sintaxe: iostat [opções]
 - c** Mostra apenas estatísticas da CPU
 - d** Mostra apenas estatísticas de I/O de disco
 - p sda** Mostra apenas estatística para sda

Monitoramento LINUX

- Comando **mpstat**

- Exibe estatísticas sobre todos os processadores existentes na máquina

Sintaxe: mpstat [opções]

- P ALL – exibir estatísticas para todas as CPUs
 - [Num] [num] – tempo de coleta dos dados e loop

Monitoramento LINUX

Comando **pidstat**

Com o pidstat podemos monitorar as informações que encontram-se no /proc/<pid>/...,

Sintaxe: pidstat [opções]

- d** estatísticas de I/O

- u** Utilização de CPU

- p** <PID> número do processo

- r** page faults e utilização da memória

[num] [num] intervalo em segundos e número de relatórios.

Monitoramento LINUX

- Comando **pidstat**

```
stress --vm 1 --vm-bytes 10M -t 300s
```

```
root@cliente-IPMH61R2:/home/cliente# pidstat -p 21671 -r 2 3
Linux 3.13.0-37-generic (cliente-IPMH61R2)    21-11-2014    _x86_64_    (4 CPU)

12:09:12      UID      PID minflt/s  majflt/s     VSZ    RSS   %MEM Command
12:09:14    1000    21671      0,00      0,00   7308    432   0,01 stress
12:09:16    1000    21671      0,00      0,00   7308    432   0,01 stress
12:09:18    1000    21671      0,00      0,00   7308    432   0,01 stress
Average:    1000    21671      0,00      0,00   7308    432   0,01 stress
root@cliente-IPMH61R2:/home/cliente#
```


Monitoramento LINUX

Comando **dstat**

Permite efetuar monitoramento e verificar performance do sistema Linux, possuindo características dos comandos **top**, **vmstat**, **free**, **iostat** combinadas.

Sintaxe: `dstat [opções]`

`Dstat n` permite ajustar o intervalo de atualização para `n` segundos

- m** uso de memória
- c** estatística de CPU
- d** Estatística de disco
- i** interrupções
- n** estatísticas de uso de rede
- fs** estatísticas do sistema de arquivos
- ntp** mostra a hora a partir de um servidor de NTP

NMON

NMON

- Nigel's performance **M**onitor;
- Por default o nmon inicia em modo live com monitoramento via terminal

```
nmon-16a-----[H for help]-----Hostname=vm26-----Refresh= 2secs -----08:53.14-----

-----
nmon-16a-----[H for help]-----Hostname=vm26-----Refresh= 2secs -----08:53.14-----
-----
For help type H or ...
nmon -? - hint
nmon -h - full details

To stop nmon type q to Quit

DISTRIB_DESCRIPTION="Ubuntu 19.04"
PowerVM POWER8 (architected), altivec supported CHRP IBM,8284-22A
PowerVM Entitlement=1.00 VirtualCPUs=2 LogicalCPUs=16
PowerVM SMT=8 Capped=0
Processor Clock=3425.000000MHz Little Endian

Use these keys to toggle statistics on/off:
c = CPU          l = CPU Long-term      - = Faster screen updates
m = Memory       V = Virtual memory  + = Slower screen updates
d = Disks        n = Network        j = File Systems
r = Resource     N = NFS              . = only busy disks/procs
k = Kernel       t = Top-processes   h = more options
q = Quit
```

NMON

- Os comandos são organizados da seguinte forma:
 - `$ nmon -f -s "seconds" -c "count"`
- Onde `-f` é salvar em arquivo (file), `-s` o intervalo entre as capturas e `-c` o número de capturas.

NMON

- O arquivo de saída do nmon é um arquivo de texto com extensão nmon e no formato CSV;
- A saída vai para o diretório atual, no seguinte formato:
 - <hostname>_<date>_<time>.nmon

NMON

- É possível especificar o local onde será armazenado o arquivo de saída através da flag -m
 - `$nmon -f -s "seconds" -c "count" -m /home/...`
- E agendar o início do experimento
 - `$ 0 0 * * * /usr/local/bin/nmon -f -s 120 -c 720 -m /home`
0 hora, 0 minutos, * dias no mês, * mês, * dia da semana.

NMON

Exemplo: Monitorar o sistema por um minuto

- 1min = 60 segundos;
- 60s com intervalos entre amostras de 5s;
- $60/5 = 12$ coletas;

Carga de trabalho:

`$stress -m 2 --vm-bytes 1G -t 50s`

Monitoramento

`$nmon -f -s 5 -c 12`

NMON

- Os dados obtidos via nmon podem ser analisados através de várias ferramentas (excel, R, Mathematica) muitos dados tornam a análise mais precisa, porém implicam em uma grande massa de dados que em sua análise tende a consumir uma grande quantidade de memória;
- developers.google.com/chart é uma biblioteca que pode facilitar um pouco se utilizada em conjunto com o nmonchart.

NMON

- 1) Instalação do nmonchart [aguther/nmonchart](https://github.com/aguther/nmonchart)
- 2) Extrair pasta nmonchart-master, instalar o ksh;
- 3) Atribuir permissões de leitura e escrita ao *build.sh*;
- 4) Executar
 - \$./nmonchart nmonfile.nmon output.html



- Monitorando Rede com TCPDUMP
 - Instalação
 - `sudo apt install tcpdump`
 - Verificar a versão
 - `tcpdump --version`
 - Sintaxe
 - `tcpdump [opções]`

Monitoramento LINUX

- Opções principais:
 - Lista de interfaces.
 - `tcpdump -D`
 - Interface específica.
 - `tcpdump -i eth0`
 - Capturar N pacotes.
 - `tcpdump -c 10 -i eth0`

Monitoramento LINUX

- Campos:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp5s0, link-type EN10MB (Ethernet), capture size 262144 bytes
04:04:16.324600 IP maria.ssh > 192.168.100.5.54566: Flags [P.], seq 4142187904:4142188092, ack 3080325120, win 501, options [nop,nop,TS val 3495424434 ecr 2420209065], length 188
04:04:16.325366 IP 192.168.100.5.54566 > maria.ssh: Flags [.], ack 188, win 24566, options [nop,nop,TS val 2420209138 ecr 3495424434], length 0
```

1. *Timestamp;*
2. *Tipo do pacote;*
3. *End.: de origem e Porta de origem;*
4. *Sinal >*
5. *End.: de destino e Porta de destino;*
6. *Bit de controle*

Monitoramento LINUX

- Opções principais:

- Mostra somente IPs.

- `tcpdump -n -i eth0`

- Grava os pacotes capturados em um arquivo.

- `tcpdump -w arquivo.pcap`

- Lê os pacotes a partir de um arquivo.

- `tcpdump -r arquivo.pcap`

Monitoramento LINUX

- Opções principais:

- Lê os pacotes a partir de um arquivo e imprime a data e hora da captura.

- `tcpdump -tttt -r arquivo.pcap`

- Imprime pacotes em ASCII.

- `tcpdump -A -i eth0`

Monitoramento LINUX

- Opções principais:

- Mostra o conteúdo do pacote em HEX e ASCII.

- `tcpdump -XX -i eth0`

- Captura pacotes originados apenas do host com o IP ou hostname.

- `tcpdump -n src host 192.168.0.1`

Monitoramento LINUX

- Opções principais:

- Captura pacotes destinado apenas ao host com IP ou hostname.

- `tcpdump -i dst host 192.168.0.1`

- Captura apenas pacotes que sejam maiores que bytes.

- `tcpdump -i eth0 greater 100`

- *captura pacotes maiores que 100 bytes.*

Monitoramento LINUX

- Opções principais:

- Captura apenas pacotes que sejam menores que bytes.

- `tcpdump -i eth0 less 100`

- Trabalha com pacotes destinados a uma porta específica.

- `tcpdump -i eth0 port 22`

Monitoramento LINUX

- Opções principais:

- Captura pacotes cujas portas estejam no intervalo entre **22** e **80**

- `tcpdump -i eth0 portrange 22-80`

- Captura somente tráfego associado ao protocolo **ICMP** na interface **eth0**

- `tcpdump -i eth0 icmp`

Monitoramento LINUX

- Opções principais:

- Captura os pacotes somente se satisfazerem às condições **A** e **B** simultaneamente.

- **tcpdump -i eth0 dst A and B**

- *por exemplo: **192.168.0.1** and **ICMP***

- Captura os pacotes que não satisfaçam à condição **A**

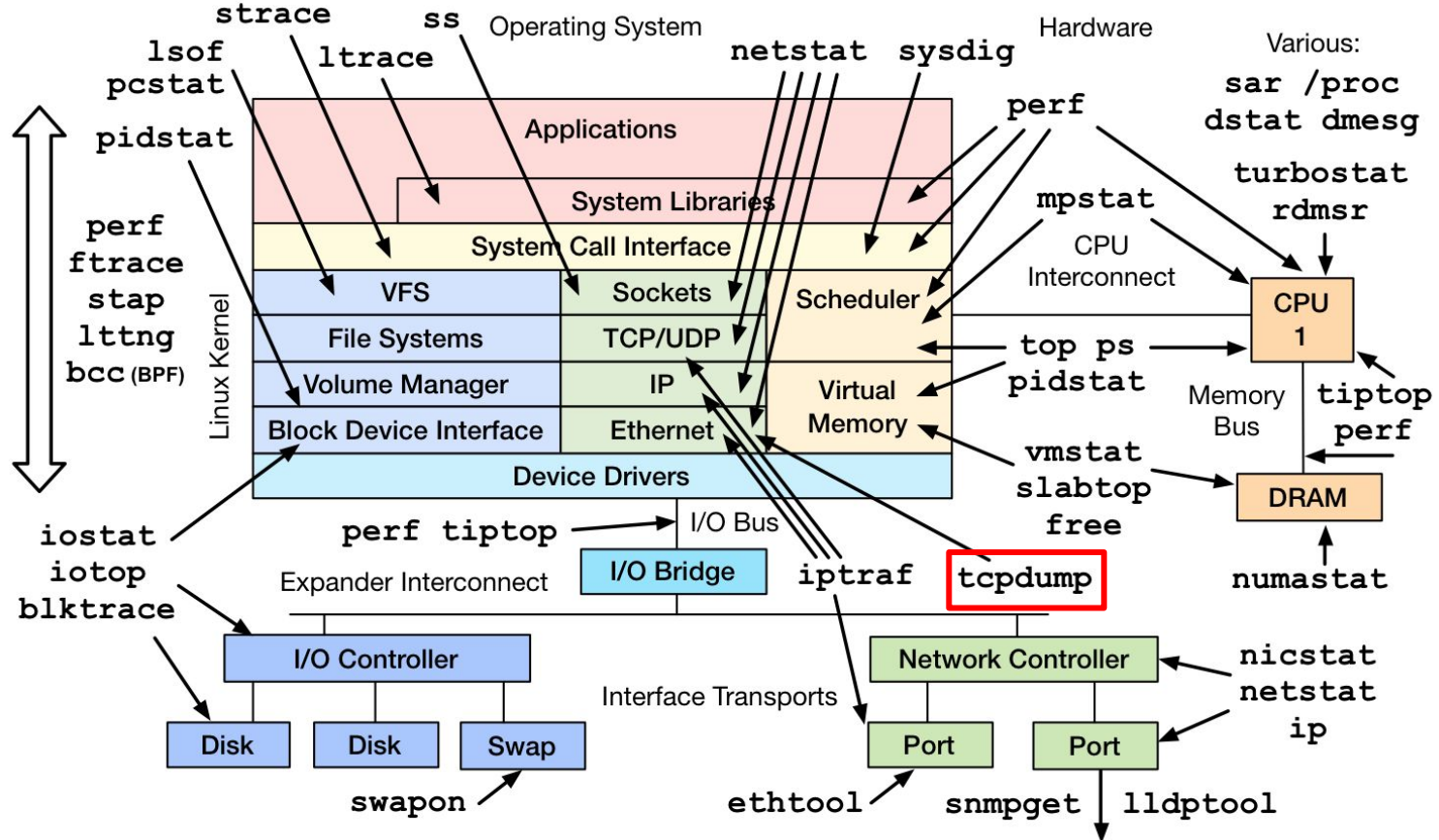
- **tcpdump -i eth0 -s 64 not A**

- *por exemplo: **not port 22***

Monitoramento LINUX

- Opções principais:
 - Outros exemplo:
 - `tcpdump -i eth1 -s 1500 port not 22 and port not 53`

Linux Performance Observability Tools





COFFEE TIME!!!

Monitoramento LINUX

- **Crontab:**

- O Cron permite que usuários do Linux e Unix executem comandos ou scripts em uma determinada data e hora. Você pode agendar scripts para serem executados periodicamente.

Monitoramento LINUX

- Opções principais:
 - Vamos abrir o terminal e digitar:
 - **crontab -e**
 - Isso significa que queremos criar um script dentro do crontab.
 - **crontab -r**
 - Deleta o script usado anteriormente.
 - **crontab -l**
 - Lista tudo.

Monitoramento LINUX

```
* * * * * comando a ser executado
- - - - -
| | | | |
| | | | ----- Dia da semana (0 - 7) (domingo = 0 ou 7)
| | | ----- Mês (1-12)
| | ----- Dia do mês (1-31)
| ----- Hora (0 - 23)
----- Minuto (0 - 59)
```

- Dado isso, montamos nossa primeira linha de comando :

■ 10 00 * * * /usr/sbin/tcpdump -n -c 30000 -w /root/debug.txt

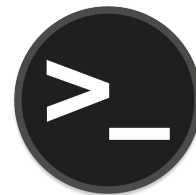
Monitoramento LINUX

Strings	Significado
@reboot	Execute uma vez, na inicialização.
@yearly	Execute uma vez por ano, "0 0 1 1 *".
@annually	(o mesmo que @yearly)
@monthly	Execute uma vez por mês, "0 0 1 * *".
@weekly	Execute uma vez por semana, "0 0 * * 0".
@daily	Execute uma vez por dia, "0 0 * * *".
@midnight	(o mesmo que @daily)
@hourly	Execute uma vez por hora, "0 * * * *".

Monitoramento LINUX

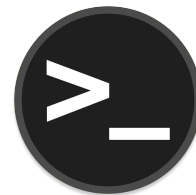
- Com isso podemos refinar ainda mais nosso crontab:
 - `@midnight /usr/sbin/tcpdump -n -c 30000 -w /root/debug.txt`
 - ou seja, 1 vez por dia nosso script vai ser criado, de acordo com os parâmetros definidos.
 - Em seguida verifique o que foi capturado.
 - `tcpdump -X -vv -r /root/port.debug.txt`

Monitoramento LINUX



- **SSH (Secure Shell)**
 - **Instalação**
 - **sudo apt install openssh-server**
 - Conexão SSH é uma forma segura de acessar remotamente outro micro, basta que você tenha habilitado o serviço SSH em ambos os micros.
 - **sudo systemctl status ssh** “verificar o status do SSH”
 - Por exemplo: **ssh user@192.168.0.1 'uptime'**
 - nos mostra a quanto tempo o computador está ligado

Monitoramento LINUX



- **SSH (Secure Shell)**
 - Para não ser redundante, vejam que alguns dos comandos mostrados anteriormente, podem ser usados no lugar do 'uptime'.
 - Também é possível executar scripts remotamente, vejamos:
 - `ssh user@192.168.0.1 ./script.sh`
 - criamos o script e damos permissão
 - `chmod +x script.sh`

Referências

- Man-pages do Linux
- MoDCS <http://www.modcs.org/>
- [Performance Tools](#)
- Advanced [TCPDUMP](#)
- Site do iostat:
 - <http://sebastien.godard.pagesperso-orange.fr>
- Jain, Raj. "The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling." *New York: John Willey* (1991).
- Lilja, David J. *Measuring computer performance: a practitioner's guide*. Cambridge University Press, 2005.