

## Conteúdo: **Criar, Monitorar e Terminar Processos**

## CONTEXTUALIZAÇÃO

- **Nesta aula teremos como objetivo aprender o seguinte:**
  - Criar Processos;
  - Monitorar Processos;
  - Terminar Processos;
  - Modificar a prioridade de processos.

## GERENCIAMENTO DE PROCESSOS

- Uma parte importante da função do administrador de sistemas é gerenciar os processos;
- Na maioria dos casos, os processos se ativarão, se executarão e terminarão sem qualquer intervenção do usuário, por que eles são gerenciados automaticamente pelo **kernel**;
- Entretanto, haverá situações em que um processo se extinguirá por algum motivo desconhecido, e precisará ser reiniciado, ou por alguma razão, precisará ser parado.

## GERENCIAMENTO DE PROCESSOS

- Alguns processos poderão consumir recursos do sistema excessivamente;
  - Nesse momento é necessária a intervenção do administrador;
  - Também é necessária outras operações, tais como: instruir um processo a ler um arquivo de configuração ou instruir um processo a ter mais prioridade.

# GERENCIAMENTO DE PROCESSOS

- Processos:
  - Todo programa, seja um comando, aplicativo ou script, que esteja rodando no sistema é um processo;
  - O Shell é um processo, e todo comando que você executa a partir do shell, inicia um ou mais de seus próprios processos (conhecidos como processos filhos).

## GERENCIAMENTO DE PROCESSOS


- **Tempo de vida:** é o tempo que um processo leva para ser executado.
  - **Processos com tempos de vida curtos:** ls;
  - **Processos com tempo de vida mais longos:** navegador WEB;
  - **Processo longos:** daemons do linux que rodam desde a inicialização.
    - Ex: syslogd é o daemon que implementa o recurso de log do sistema .
  - Quando um processo termina, ele extinguiu ou morreu.

## GERENCIAMENTO DE PROCESSOS

- **ID ou PID do processo:**
  - Número do processo, atribuído quando ele inicia;
  - São números inteiros, que não se repetem.
- **ID/PID do usuário e do Grupo (GID):**
  - Os processo precisam ter privilégios associados;
  - O PID do processo está associado ao PID e o GID do usuário que iniciou o processo;
  - Isso limita o acesso que o processo terá ao sistema de arquivos.

# GERENCIAMENTO DE PROCESSOS

- **ID do Processo Parente**

- O primeiro processo iniciado pelo kernel, no momento da inicialização do sistema é um programa chamado **INIT (systemd)** (tem o **PID 1**);  
 Nome do processo INIT no Linux.
- É o ascendente comum de todos os outros processo do sistema;
  - O Shell é um descendente do INIT;
- Todos os processo iniciados pelo SHELL são descentes ou parentes do SHELL;
  - Se o processo parente tiver desaparecido (por exemplo o do shell), o PID parente será 1, que é o PID do INIT.



# MONITORAMENTO DE PROCESSOS

- **Monitoração de processo**
  - A monitoração de processos é feita utilizando **três utilitários**:
    - **ps**,
    - **Pstree**;
    - **Top**.
- **Comando “ps”**
  - Sintaxe: `ps [opções]`
- **Descrição:** gera o status instantâneo dos processos atuais na saída padrão.

# MONITORAMENTO DE PROCESSOS

- **Opções mais frequentes:**

- **-a:** mostra os processos que são propriedades de outros usuários e estão vinculados a um terminal, além dos processos do usuário atual;
- **-u:** inclui o nome do usuário e a data de início do processo;
- **-x:** inclui processos que não tenham terminais de controle (shell).
  - Essa opção é frequentemente necessária para se ver processos deamons e outros que não tenham sido iniciados a partir de uma sessão de terminal.




Fazer um exemplo com cada um dos argumentos junto com o professor.

No fim, use o `ps -aux | more` e veja o PID dos processos

# MONITORAMENTO DE PROCESSOS

- **pstree**
  - **Sintaxe:** pstree [opções] [pid/usuário]
  - **Descrição:** exibe uma lista hierárquica de processos em um formato de árvore
    - Bastante útil para entender a relação de processos parentes/descendentes.
  - Se o PID for especificado, a árvore exibida tem a sua raiz no processo em questão.
  - Se o nome de usuário for especificado, serão exibidas todas as árvores para todos os processos de propriedade desse usuário.

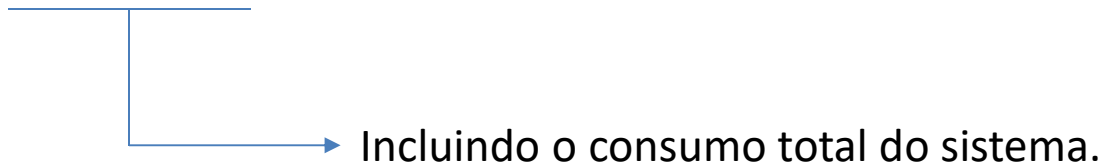


Fazer um exemplo com cada um dos argumentos junto com o professor, utilizando Cada uma das opções, mas antes instalar via apt o utilitário **psmisc** que contém o **pstree**.

**Ex: pstree joilson**

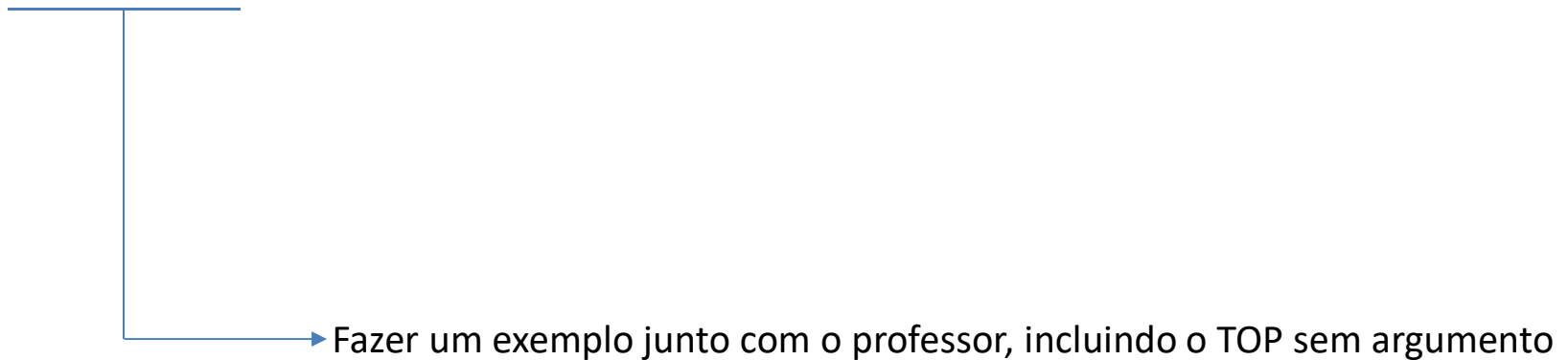
# MONITORAMENTO DE PROCESSOS

- **TOP**
- **Sintaxe:** top [opções]
- **Descrição:** exibe o status dos processos continuamente atualizados, com consumo da CPU, memória.



# MONITORAMENTO DE PROCESSOS

- **-b:** Resultado enxuto
  - Ex: `top -b`
- **-d [tempo]:**
  - **Ex: `top -d 2` (Tempo de atualização (a cada 2 segundos))**



# MONITORAMENTO DE PROCESSOS

- **Opções interativas utilizadas com o TOP**
  - Uma vez que o top esteja rodando interativamente, ele pode receber uma série de comandos, com por exemplo:
    - h: Gera uma tela de ajuda
    - k: Termina processos. Será pedido o PID do processo e o sinal para enviar para ele (15 SIGTERM)
    - 1: mostra todas as CPUs
    - q: sai do programa
    - u: mostra processos de determinado usuários



Fazer exemplos com o professor.

## ENVIANDO MENSAGENS AOS PROCESSOS

- **Kill**
  - O kill é usado para enviar mensagens aos processos.
- Sintaxe
  - `kill [signum] [pids]`
- Signums:
  - 15: tenta finalizar um processo de forma elegante, pede que um processo finalize sem causar problema;
  - 9: finaliza imediatamente e incondicionalmente o processo (medida drástica, não pode ser ignorado pelo processo).

## ENVIANDO MENSAGENS AOS PROCESSOS

- **Killall: mata processo pelo nome**
  - killall [nome do processo]
- **Exemplo:**
  - killall apache2
  - killall -9 apache2



## ENVIANDO MENSAGENS AOS PROCESSOS

- **Killall: mata processo pelo nome**

- killall [nome do processo]

- **Exemplo:**

- killall apache2
  - killall -9 apache2



Junto com o professor, abra dois terminais, inicie um editor de arquivos em um dos terminais e no outro terminal finalize o processo com o editor.

## CONTROLANDO TAREFAS NO SHELL

- **Controle das Tarefas do Shell**

- É a habilidade do shell (com apoio do kernel) de interromper e retomar a execução de comandos, bem como colocá-los no background;
- Diz-se que o processo está em **foreground** se ele está executando em seu terminal;
- Quando ele está em **background**, o processo não aparece no terminal do usuário, mas existe a possibilidade de enviar sinais ao processo;
- A principal razão para se ter um processo em background e deixar o terminal do usuário livre;

## CONTROLANDO TAREFAS NO SHELL

- Colocando um processo em background na inicialização:
  - Ex: `vim teste &` (o `&` é o sinal)
- Colocando um processo em background após a inicialização:
  - `$ "Ctrl Z"`
- Verificando todos os processo em background
  - `$ jobs`

## CONTROLANDO TAREFAS NO SHELL

- Colocando um processo em foreground
  - **fg** [jobspec]
- Colocando um processo em background
  - **bg** [jobspec]

# MODIFICANDO PRIORIDADE DE EXECUÇÃO DOS PROCESSOS

- **Modificar prioridade de execução de processos**
  - É possível alterar a prioridade de execução dos processos através dos comandos **nice** e **renice**.
  - O comando **nice** ajusta o tempo disponível de CPU de um processo para mais ou para menos prioridade na inicialização do processo.
  - Se o ajuste de prioridade para um processo for um número positivo, quer dizer que ele está sendo mais legal com os outros programas diminuindo a sua prioridade.
  - Se o ajuste for um número negativo, quer dizer que o programa está sendo menos legal, aumentando a sua prioridade de execução e sobrando menos tempo de CPU para os outros programas.

# MODIFICANDO PRIORIDADE DE EXECUÇÃO DOS PROCESSOS

- O ajuste de prioridade possível vai do -20 (mais prioridade / menos legal) até o 19 (mais legal, menos prioridade).
- Se não for passado nenhum valor de ajuste, o comando nice ajustará a prioridade para **+10**, diminuindo o tempo de execução do processo (por padrão a maioria dos processo são iniciados com 0).
- **\$ nice updatedb &**
  - Neste exemplo o comando updatedb tem menos prioridade de execução.
- **\$ nice -n -10 folha\_pagamento**
  - Neste exemplo o comando folha de\_pagamento será executado com mais prioridade.



Junto com o professor, inicie a edição de um arquivo com prioridade 10 e depois -10 (como root).

Em um outro terminal, veja a prioridade se alterando.

Obs: capture o numero do processo com **ps -aux** , depois com o **top** veja somente esse processo: **top -p <numero do processo>**

# MODIFICANDO PRIORIDADE DE EXECUÇÃO DOS PROCESSOS

- **Comando renice**

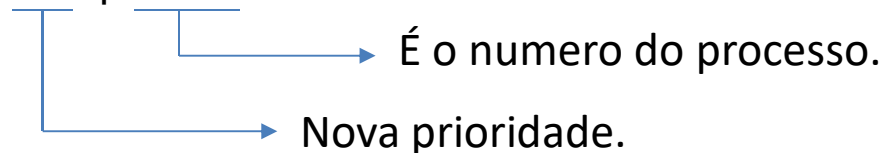
- Uso:
- `$ renice [+/-] ajuste_prioridade [opções] PID/Usuário`
- O comando `renice` ajusta a prioridade de execução de processos **que já estão rodando**. Por padrão, o comando `renice` recebe como parâmetro o PID de um determinado processo.
- O ajuste de prioridade é um número inteiro que vai do `-20` (maior prioridade) até o `+20` (executar qualquer coisa antes deste processo).

# MODIFICANDO PRIORIDADE DE EXECUÇÃO DOS PROCESSOS

- **As opções mais usuais são:**

- **-p:** Recebe um **PID** para alterar sua prioridade.

- Ex: `renice -15 -p 2806`



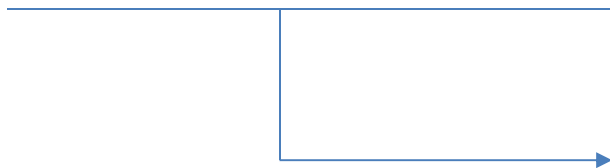
- **-u:** Recebe um **nome de usuário** para alterar a prioridade de todos os processos deste usuário em execução.

- **-g:** Recebe um **nome de um grupo** para alterar a prioridade de todos os processos pertencentes a este grupo



## MODIFICANDO PRIORIDADE DE EXECUÇÃO DOS PROCESSOS

- Exemplo:
- `# renice -1 -u root`
  - Neste exemplo, todos os processos de que o usuário root é dono terá mais prioridade.



Junto com o professor, inicie a edição de um arquivo, depois altere a prioridade em tempo de execução.

Obs: capture o numero do processo com `ps -aux` , depois com o `top` veja somente esse processo: `top -p <numero do processo>`

## EXERCÍCIOS

- 1. Com o "ps" verifique os processos de todos os usuários vinculados a um terminal com seus respectivos nomes;
- 2. Com o "ps" verifique os processos de todos os usuários com seus respectivos nomes, inclusive aqueles que não tenham sido vinculados a um terminal;
- 3. Com o "ps" verifique os processos de todos os usuários vinculados a um terminal com seus respectivos nomes, além disso filtre somente os processos do seu usuário;
- 4. Exiba uma lista hierárquica de processos em um formato de árvore (todos do sistema);
- 5. Exiba uma lista hierárquica de processos em um formato de árvore do seu usuário;

## EXERCÍCIOS

- 6. Exiba o status dos processos continuamente atualizados, com consumo da CPU, memória;
- 7. Exiba o status dos processos continuamente atualizados, com consumo da CPU, memória atualizados a cada 1s;
- 8. Exiba o status dos processos continuamente atualizados e interativamente veja todas as CPUs do sistema;
- 9. Crie um processo, em um terminal e em outro envie uma mensagem para que o processo seja terminado imediatamente, de forma drástica;
- 10. Crie três processo em foreground em seu terminal, coloque todos em background, verifique o ID de todos e coloque o terceiro em foreground novamente.
- 11. Atribua prioridade máxima para todos os processos do seu usuário.
- 12. Crie um processo e mude a prioridade dele para -1 em tempo de execução. Além disso, verifique e comprove se a prioridade realmente foi alterada.

**Perguntas ?**