Linux Commands

Tiago Heinrich

UniSociesc Joinville

19/03/2020

GCC

- GNU Compiler Collection (GCC) é um sistema compilador produzido pelo Projeto GNU que suporta várias linguagens de programação
- C (gcc), C++ (g++), Objective-C, Objective-C++, Fortran (gfortran), Ada (GNAT), Go (gccgo), OpenMP, D e OpenACC
- C++14 e C++17

GCC

```
gcc -c hello.adb
gnatbind hello.ali
gnatlink hello
./hello

gcc main.c -o out
./out
```

Execução de programas

- Para iniciar um programa em primeiro plano basta digitar o seu comando num terminal
- Para executa-lo em segundo plano, deve-se adicionar **&** no final dele
- Para executar comandos em sequência, basta separa-los por **;**

```
apt-get update; apt upgrade
evince arquivo.pdf &
```

Execução de programas

- Usando **&&** o comando a seguir só é executado se o primeiro foi executado sem erros
- Para ver quais os programas que estão rodando, usa-se o comando **ps**
- **top** é um comando semelhante ao **ps**. Porém ele exibe em tempo real as informações sobre os processos

Execução de programas

- Para encerrar um processo em execução em primeiro plano basta usar o atalho **ctrl+c**
- É possível também parar momentaneamente um processo usando o atalho **ctrl+z**
- O comando **jobs** mostra os processos parados ou rodando em segundo plano.

Execução de programas

A execução de um programa pode ocorrer em dois modos:

- Primeiro plano: Também chamado de foreground (**fg**), é a forma na qual você deve esperar que o programa seja encerrado para poder rodar um novo comando
- Segundo plano: Também chamado de backgroud (**bg**), o programa é executado em segundo plano e deixa o terminal livre

Execução de programas

Como restaurar um programa parado?

- fg [#] - Faz um programa em segundo plano ou parado rodar em primeiro plano
- bg [#] - Faz um programa parado continuar a execução em segundo plano
- # = Número do processo obtido com o comando **jobs**

Execução de programas

Como restaurar um programa parado?

- Para se matar (finalizar) um processo pode-se usar o comando **kill** ou **killall**.

```
kill [PID]
killall nome\_processo
```

Execução de programas

- É possível fazer o controle de prioridade de execução do processo usando o comando **nice**.

  nice [−n −#] [*comando/programa*]

- Onde *n* é a prioridade, que varia de 19 (mínimo) até -20 (máximo)

init

- *init* (abreviação de inicialização) é o primeiro processo iniciado durante a inicialização do sistema do computador. É um daemon que continua em execução até o sistema ser desligado
- Em qualquer momento o sistema esta em um dos números predeterminados de estados, chamados de *runlevels*. Pelo menos um nível de execução é o estado operacional normal do sistema

runlevel

- Nível de execução do sistema. Define o estado da máquina após a inicialização
- O runlevel indica o modo de operação atual da máquina, definindo quais serviços e recursos devem permanecer ativos.
- São sete níveis de execução numerados de 0 a 6

runlevel

- 0 - Interrompe a execução do sistema. todos os programas e daemons finalizados. É acionado pelo comando shutdown -h
- 1 - Modo monousuário, modo para tarefas administrativas (útil para manutenção dos sistema)
- 2 - Modo multiusuário, não configura interfaces de rede e não exporta serviços de rede
- 3 - Modo multiusuário, inicia o sistema normalmente
- 4 - Modo multiusuário, para fins especiais
- 5 - Modo multiusuário com login gráfico (semelhante ao modo 3)
- 6 - Reinicialização do sistema. Todos os programas e daemons são encerrados e o sistema é reiniciado. É acionado pelo comando shutdown -r

Syslog

- São arquivos onde é gravado atividades de programas. Encontrado em */var/log*
- Permite a separação do *software* que gera mensagens, do sistema que as armazena e do *software* que as relata e analisa. Cada mensagem é identificada com um código de instalação, indicando o tipo de *software* que está gerando a mensagem e atribuído um nível de gravidade
- O syslog pode ser utilizado para gerenciamento de sistema e auditoria de segurança, bem como mensagens gerais de informação, análise e depuração.

Severity level

0. *emerg* (Emergência) O sistema está inutilizável (condição de pânico)
1. *alert* (Alerta) A ação deve ser tomada imediatamente
2. *crit* (Crítico) Condições críticas
3. *err* (Erro) Condições de erro
4. *warning* (Atenção) Condições de aviso
5. *notice* (Aviso) Condições normais, mas significativas
6. *info* (Informativo) Mensagens informativas
7. *Debug* (Depurar) Mensagens no nível de depuração

Severity level

```
import syslog
import logging

my_logger.debug('Syslog class run function')
return my_logger

except Exception as a:
    my_logger.critical('syslog class error', a)
```

Severity level

- auth - Mensagens de segurança/autorização (é recomendável usar authpriv ao invés deste)
- authpriv - Mensagens de segurança/autorização (privativas)
- cron - Daemons de agendamento (cron e at)
- daemon - Outros daemons do sistema que não possuem facilidades específicas
- kern - Mensagens do kernel
- security - Sinônimo para a facilidade auth (evite usa-la)
- syslog - Mensagens internas geradas pelo syslogd

Regex

- Unix text processing cap 11
- Existem aplicativos diferentes que usam diferentes tipos de regex no Linux, como linguagens de programação (Java, Perl, Python ,,,) e programas Linux como (sed, awk, grep)

awk

- Linguagem de digitalização e processamento direcionada a padrões
- O *awk* varre cada arquivo de entrada e busca por linhas que correspondam a qualquer conjunto de padrões especificado

```
awk '/\$/{print $0}' myfile
echo "" | awk '/\$/{print $0}'
cat myfile | awk '/\$/{print $0}'
```

awk

- Os padrões usam caracteres especiais. E você não pode incluí-los em seus padrões e, se o fizer, não obterá o resultado esperado.
- *Case sensitive*
- Espaços ou números

```
.*[]^${}\+?|()

echo "Hello friend" | awk '/Friend/{print $0}'
echo "Hello friend" | awk '/friend/{print $0}'

echo "Testing regex 2 again" | awk '/regex 2/{print $0}'

echo "\$){\\" | awk '/\$\)\{\\/{print $0}'
echo "1/2" | awk '/\//{print $0}'
```

awk

- Localizar o início de uma linha
- Localizar no final de uma linha

```
1 echo -e "ola \nNome: Tiago\nTiago e meu Nome" | awk '/^Nome/{print $0}'
2
3 echo "Agora no meio ^ do texto" | awk '/\^/{print $0}'
4
5 echo -e "ola \nNome: Tiago\nTiago e meu Nome" | awk '/Nome$/{print $0}'
```

awk

- Você pode combinar qualquer caractere com o caractere especial de ponto
- A classe de caracteres corresponde a um conjunto de caracteres, se algum deles foi encontrado, o padrão corresponde
- Procurando por um caracteres que não está na classe

```
echo -e "ola\nNome Tiago\nTIago Super Nome\nagoTI" |
awk '/T.ago/{print $0}'

echo -e "ola\nNome: Tiago\nTiago e meu Nome" | awk '/^[NT]/{print $0}'

echo -e "ola\nNome: Tiago\nTiago e meu Nome" | awk '/^[^NT]/{print $0}'
```

awk

- Para especificar um intervalo de caracteres, você pode usar o símbolo (-)

```
echo -e "def function\nint function\ndef new function\n" |
awk '/[d-f]\ /{print $0}'

echo -e "19000\n 1000\n 100\n 10" | awk '/[1-9][1-9]000/{print $0}'
```

awk

- O asterisco significa que o personagem deve existir zero ou mais vezes
- O ponto de interrogação significa que o caractere anterior pode existir uma ou nenhuma.
- O sinal de adição significa que o caractere antes do sinal de adição deve existir uma ou mais vezes, mas deve existir pelo menos uma vez
- Chaves permitem especificar o número de existência de um padrão

```
echo -e "19000\n 1000\n 100\n 10" | awk '/[1-9][1-9]*0/{print $0}'
echo -e "19000\n 1000\n 100\n 10" | awk '/[1-9]0?/{print $0}'
echo -e 'Tiag\nTiagoooo\nTIago' | awk '/ago+/{print $0}'
echo -e 'Tiag\nTiagoooo\nTIago' | awk '/ago{1,4}/{print $0}'
```

awk

- O símbolo do pipe cria um *OR* lógico entre 2 padrões. Se um dos padrões existir, ele será bem-sucedido; caso contrário, ele falhará
- Você pode agrupar expressões para que os mecanismos regex as considerem uma peça

```
echo -e 'ola meu nome\nola seu nome\nrandom random nome' |
awk '/ola\ meu|seu/{print $0}'
```

sed

- **sed** stream editor
  - **g** substitua todas as ocorrências
  - **A** o número da ocorrência do novo texto que você deseja substituir
  - **p** imprima o conteúdo original
  - **w** significa escrever os resultados em um arquivo
  - **d** remover elementos

```
1 echo "Super name Tiago" | sed 's/Tiago/Rodrigo/'
2
3 echo -e 'Ola tiago\nOla rodrigo\nOla victor' > myfile
```

sed

```
sed 's/Ola/Tchau/g' myfile
sed 's/Ola/Tchau test/w output' myfile
sed '2s/Ola/Tchau/g' myfile
sed '2,3s/Ola/Tchau/g' myfile
sed '1,$s/Ola/Tchau/g' myfile
sed '2d' myfile
sed '/tiago/d' myfile
sed '/rodrigo/,/victor/d' myfile
```

Próxima aula

- sed & awk
- Virtual Box
- Install Ubuntu Server
- `https://ubuntu.com/download/server`