105 Shells e scripts do Shell

105.1 Personalizar e usar o ambiente do shell

Lição 1

Tipos de shell: Interativo x Não-interativo e Login x Sem login

Shells interativos / não-interativos: Este tipo de shell refere-se à interação que ocorre entre o usuário e o shell: O usuário fornece a entrada digitando comandos no terminal com o teclado; o shell fornece a saída imprimindo mensagens na tela.

Shells com login / sem login: Esse tipo de shell se refere ao caso de um usuário que acessa um sistema fornecendo suas credenciais, como nome de usuário e senha.

Maneiras de iniciar diferentes shells

sudo su - rafael, sudo su -l rafael ou sudo su --login rafael - iniciam um shell de login interativo como rafael.

sudo su rafael, sudo -u rafael -s - inicia um shell sem login interativo como rafael.

sudo su - root ou sudo su - inicia um shell de login interativo como root.

sudo su root ou sudo su inicia um shell sem login interativo como root.
sudo -s ou sudo -u root -s iniciam um shell sem login como root.

sudo -i inicia um shell de login interativo como root.

sudo -i <algum_comando> inicia um shell de login interativo como root, executa o comando e retorna ao usuário original.

Arquivos:

- Scripts globais que configuram o ambiente global (de todos os usuários) em /etc >> /etc/profile e /etc/bash.bashrc
- Carrega o perfil desse arquivo para todos os usuários: /etc/profile
- Arquivo que será carregado em todos os casos (ao executar shell interativo com/sem login com user ou root_ - /etc/bash.bashrc
- Em sequência: ~/.bashrc - Arquivo próprio de cada usuário, caso houver. Se não, executa ~/.profile

su -l rafael - simula bash com login (isso faz executar todos os arquivos nesta sequência: /etc/bash.bashrc → /etc/profile → ~/.bashrc → ~/.profile

---

Lição 2

Regras para variáveis
Nomes de variáveis

O nome da variável pode conter letras (a-z,A-Z), números (0-9) e sublinhados (_):

Valores das variáveis

As variáveis podem conter quaisquer caracteres alfanuméricos (a-z,A-Z,0-9), além da maioria dos outros caracteres (?,!,*,.,/, etc.)

- O conteúdo não pode conter ! no meio, ele pode ser usado somente no final
- É necessário usar “” ou ‘’ para poder utilizar espaços, símbolos de <,>, | substituição de variáveis (este funciona somente com aspas duplas, já que aspas simples interpreta todos os caracteres literalmente)
- Todas as barras invertidas devem ser escapadas com outra barra invertida. Aliás, se uma barra invertida for o último caractere na string e não o escaparmos, o Bash interpretará que queremos uma quebra de linha e criará uma nova linha

Variáveis locais ou do Shell

Variáveis locais ou de shell existem apenas no shell em que foram criadas. Por convenção, as variáveis locais são escritas em letras minúsculas.

Variáveis globais ou de ambiente

Existem variáveis globais ou de ambiente para o shell atual, bem como para todos os processos subsequentes gerados a partir dele. Por convenção, as variáveis de ambiente são escritas em letras maiúsculas

export ou **export -p**- mostra uma lista de todas as variáveis de ambiente existentes
export variable_name - torna variável global
export -n variable_name - variável novamente transformada em variável local

printenv e **env** - usados para imprimir uma lista de todas as variáveis de ambiente
printenv pode ser usado para verificar o valor de uma variável de forma semelhante à echo (sem o $)

Executando um programa em um ambiente modificado

env -i bash - cria uma nova sessão bash com o ambiente o mais vazio possível, removendo a maioria das variáveis (além de funções e aliases)

env -u variável - remove a variável do ambiente

Em shells não interativos sem login, vimos como os scripts não lêem nenhum arquivo de inicialização padrão, mas, em vez disso, procuram o valor da variável BASH_ENV e a usam como arquivo de inicialização, se ela existir.

Variáveis de ambiente comuns
$HOME
$DISPLAY
$HISTCONTROL
$HISTSIZE
$HISTFILESIZE
$HISTFILE
$HOSTNAME - Esta variável armazena o nome TCP/IP do computador hospedeiro:
$HOSTTYPE - Armazena a arquitetura do processador do computador hospedeiro:
$LANG - Idioma do sistema
$LD_LIBRARY_PATH - Esta variável consiste em um conjunto de diretórios separados por dois pontos nos quais as bibliotecas compartilhadas são compartilhadas por programas
$MAIL - Esta variável armazena o arquivo no qual o Bash procura por email
$MAILCHECK - Esta variável armazena um valor numérico que indica, em segundos, a frequência com que o Bash verifica se há novas mensagens
$PATH - Esta variável de ambiente armazena a lista de diretórios onde o Bash procura por arquivos executáveis quando instruído a executar qualquer programa. Em nossa máquina de exemplo, esta variável é definida através do arquivo de sistema /etc/profile
$PS1 -** Essa variável armazena o valor do prompt do Bash. ID root = 0, PS1='#'. Outros usuários PS1=$. Definida no /etc/profile. Ver ID do usuário: **id -u

PS2 - Normalmente definido como > e usado como prompt de continuação para comandos longos de muitas linhas.
PS3 - Usado como prompt para o comando select.
PS4 - Normalmente definido como + e usado para depuração.

$SHELL - Armazena o caminho absoluto do shell atual
$USER - Armazena o nome do usuário atual

Lição 3 - Aliases e funções

alias
Um alias é um nome substituto para outro(s) comando(s). Ele pode ser executado como um comando normal, mas, na verdade, executa outro comando de acordo com a definição do alias.

Estrutura: **alias alias_name=command(s)**

Para criar alias com o comando LL, por exemplo - **alias ll='ls -lha'**
Comando unalias para remover alias

Usar aspas simples ou dupla quando houver espaços, pode-se referenciar variáveis dentro do alias. Para escapar um alias, usa \. Pode-se usar um alias dentro de outro alias. pode-se colocar uma função dentro de um alias

Expansão e avaliação de aspas em aliases

Ao se usar aspas com variáveis de ambiente, as aspas simples tornam a expansão dinâmica. No entanto, com aspas duplas, a expansão é feita estaticamente: **alias where?='echo $PWD'** e **alias where?="echo $PWD"**

Persistência de aliases: scripts de inicialização

Colocar os aliases no arquivo ~/.bashrc, ou então no arquivo que será lido por .bashrc, o ~/.bash_aliases

---

fuction
Para criar funções, duas sintaxes são possíveis:

Usando a palavra-chave function:
function function_name {
}

Usando ()
function_name() {
}

Podem ser escritas diretamente no prompt do shell (com a varíavel PS2 indicando as quebras de linhas) ou em arquivos e scripts. Pode-se digitar em várias linhas, ou em uma única utilizando ponto e vírgula (depois do último comando também tem ponto e vírgula)

Para que as funções sejam persistentes durante as reinicializações do sistema, temos de colocá-las em scripts de inicialização do shell, como **/etc/bash.bashrc** (global) ou **~/.bashrc** (local).

Exemplo de função para criar usuário:

```
function criausuario () {
echo “Nome”
read NOME
echo “Login”
read LOGIN
useradd -m -d/home/$LOGIN -c”NOME” -s/bin/bash $LOGIN
passwd $LOGIN
}
```

Depois de adicionar aliases ou funções para qualquer arquivo de script de inicialização, é preciso executar . ou source nesses arquivos para que as alterações tenham efeito (caso você não queira fazer logout e login novamente ou reinicializar o sistema).

Variáveis integradas especiais do Bash

São especiais porque só podem ser referenciadas — e não atribuídas

Variáveis especiais

$? - Recupera o valor de retorno do último comando. 0 → sucesso. Diferente de zero → erro

$$ - retorna número do PID do shell atual (ID do processo)

$! - retorna PID do último trabalho em segundo plano

$0 - Retorna o nome do script que está sendo executado. Ex: -bash ou bash

$1-$9 - Retorna os parâmetros passados na linha de comando

${10}, ${11} ... - Retorna os parâmetros passados na linha de comando

$# - Retorna o número de parâmetros passados para o comando

$* - Expandem-se para os argumentos passados para o comando.

$@ - Mesmo que $*. Se usado com aspas duplas, como em "$@", todos os argumentos serão colocados entre aspas duplas.

$_ - Expande-se para o último parâmetro ou o nome do script

Variáveis em funções

É possível utilizar tanto variáveis locais, quanto variáveis ambientais dentro da função

Ex: Arquivo **funed**

```
editors() {
editor=emacs
echo "The text editor of $USER is: $editor."
}
editors
```

Funções em scripts

As funções são encontradas principalmente em scripts do Bash.

Uma função dentro de um alias

Ex → alias great_editor='gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }; gr8_ed'

O comando **unset** também pode ser usado para remover as funções:

unset -v → para as variáveis
unset -f → para as funções

Se usado sem opções, o unset tenta primeiro remover uma variável e — se falhar — ele tenta remover uma função.

Uma função dentro de uma função

É possível usar uma função dentro de outra função. Ex:

```
func2 () {
}
func1 () {
comandos...
func2
}
func1
```

source, ou simplesmente, **. -** incorpora um arquivo dentro de outro arquivo

Tópico 105.2 Personalizar ou criar scripts simples
Lição 1

Diretiva **#!** - indica qual é o interpretador padrão que será usado para ler e interpretar o script
#! - chamados de shebang

Os scripts são executados em um novo processo do Bash, chamado sub-shell. Isso evita que o script sobrescreva as variáveis de ambiente da sessão atual e faça modificações indesejadas nela. Se o objetivo é executar o conteúdo do script na sessão atual do shell, ele deve ser executado com source script.sh ou . script.sh (note que há um espaço entre o ponto e o nome do script).

Para que o shell atual também se encerre quando o script for concluído, o script — ou qualquer outro comando — pode ser precedido pelo comando exec

---
Variáveis

Variáveis especiais
$? - Recupera o valor de retorno do último comando. 0 → sucesso. Diferente de zero → erro
$$ - retorna número do PID do shell atual (ID do processo)
$! - retorna PID do último trabalho em segundo plano
$0 - Retorna o nome do script que está sendo executado. Ex: -bash ou bash
$1-$9 - Retorna os parâmetros passados na linha de comando
${10}, ${11} ... - Retorna os parâmetros passados na linha de comando
$# - Retorna o número de parâmetros passados para o comando
$* - Expandem-se para os argumentos passados para o comando.
$@ - Mesmo que $*. Se usado com aspas duplas, como em "$@", todos os argumentos serão colocados entre aspas duplas.
$_ - Expande-se para o último parâmetro ou o nome do script

Variáveis comuns
Têm como função armazenar valores inseridos manualmente ou a saída gerada por outros comandos.
Comando **read** → solicitar informações ao usuário.
Ex → **read NAME SURNAME**
read -p "Type your first name and last name:" NAME SURNAME

Pode-se usar a substituição de comandos para definir variáveis
O comprimento de uma variável, ou seja, a quantidade de caracteres que ela contém, é retornado acrescentando-se um hash **#** antes do nome da variável. Esse recurso, no entanto, requer o uso das chaves para indicar a variável.

Criar matrizes
As matrizes devem ser declaradas com o comando interno do Bash **declare** → ***declare -a SIZES***

Ou a partir de uma lista predefinida de itens →
lista=(gasolina alcool diesel gnv)
echo ${lista[*]}

***Número respectivo do item, a partir do 0**

for item in ${lista[*]} ; do echo “Combustível: $item"; done

- A alteração do conteúdo de um elemento da matriz é realizada assim: **lista[0]=etanol**
  SIZES[0]=1048576

- O comprimento de um elemento em uma matriz é retornado com o caractere hash: **${#SIZES[0]}**

- O número total de elementos em uma matriz é retornado se @ ou * forem usados como o índice: **echo ${#SIZES[@]}** ou **echo ${#SIZES[*]}**

As matrizes também podem ser declaradas usando-se, como elementos iniciais, a saída de um comando, por meio da substituição de comando.
Ex → **FS=($(cut -f 2 < /proc/filesystems))**

O Bash trata cada caractere do $IFS (Input Field Separator ou separador de campos) de uma variável de ambiente como um delimitador. Para alterar o delimitador de campo apenas para caracteres de nova linha, por exemplo, a variável IFS deve ser redefinida com o comando IFS=$'\n'

Expressões aritméticas

Soma → Pode ser feito com o comando **expr** ou com $(( ))
Exemplo → **SUM=`expr $VAL1 + $VAL2`** ou **SUM=$(($VAL1 + $VAL2))**

Execução condicional

Usar os comandos **if** e **test** para testar condições. Deve terminar com **fi**
Exemplo:

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
else
  echo "No, /bin/bash is not executable."
fi
```

Saída do script

Todo comando ao ser executado define um código de execução (**exit code)**:
1 - comando falhou
0 - comando executado com sucesso

/bin/false - código de execução 1
/bin/false - código de execução 0
&& - Só executará o comando seguinte do &&, se o comando anterior do && retornar 0, ou seja, tiver sucesso
|| - Só executará o comando seguinte do &&, se o comando anterior do && retornar 1
; - Executa o segundo comando independentemente do resultado do primeiro

Ex: **/bin/false && echo "Mostrar na tela" -**
Customizar o shell
echo $SHELLOPTS - mostra todas as opções possíveis de serem utilizadas para customizar o shell
shopts - mostra as opções de customização ativas no shell

Para a prova, precisa saber do **noclobber**

Lição 2

Comando test

É uma grande ferramenta para testar arquivos, permissões, textos, números, etc. Fornece o resultado do teste através da variável de retorno $?

Condições possíveis

- -eq Igual
- != Diferente
- -gt Maior

- -lt Menor
- -d Se o arquivo for um diretório
- -e Se existir o arquivo
- -z Se o arquivo estiver vazio
- -f Se o arquivo contiver algum texto
- -o Se o usuário for o dono do arquivo
- -r Se o arquivo pode ser lido
- -w Se o arquivo pode ser alterado
- -x Se o arquivo pode ser executado

Para fazer comparação com números:

- -eq : (equal) Igual à
- -ne : (not equal) Diferente de
- -lt : (less than) Menor que
- -gt : (greater than) Maior que
- -le : (less or egual) Menor ou igual à
- -ge : (greater or equal) Maior ou igual à

Para fazer comparação com textos:

- **=** : Igual à (isso mesmo apenas um sinal de igual)
- **!=** : Diferente de
- **-n** : String existe e não é vazia (apenas um operador)
- **-z**: String existe e é vazia (apenas um operador)

Para arquivos e diretórios:

- **-s**: Arquivo existe, não vazio (apenas um operador)
- **-f**: Arquivo existe, não é um diretório (apenas um operador)
- **-d**: Diretório existe (apenas um operador)
- **-w**: Arquivo, com permissão de escrita (apenas um operador)
- **-r**: Arquivo, com permissão de leitura (apenas um operador)
- **-x**: Arquivo, com permissão de execução -x (apenas um operador)

Construção condicional - case
Estrutura:

```
case $VARIAVEL in
padrão1)
comandos
;;
padrão2)
comandos
;;
```

*)
comandos
;;
esac

Construções de loop

O Bash tem três instruções de loop — for, until e while — projetadas para construções de loop ligeiramente distintas.

for

1 - *for VARNAME in LIST*
do
    COMMANDS
done

2- *for ((CONDIÇÃO))*
do
    COMMANDS
done

until

until [CONDIÇÃO]
do
    COMMANDS
done

106 Interfaces de usuário e áreas de trabalho

Um Gerenciador de Janelas (Window Manager) realiza a interação entre janelas, aplicações, e o Sistema de Janelas (Windowing System)
O sistema de janelas manipula dispositivos de hardware (mouse, placas de vídeo), assim com o posicionamento do ponteiro

Sistema de janelas - componente da GUI que suporta a implementação de gerenciadores de janelas e fornece suporte a hardware gráfico, dispositivos apontadores e teclados, não inclui as janelas em si. Ele implementa renderização de fontes, desenhos primitivos (linhas e traços) e habilita o computador a trabalhar com vários programas simultaneamente ao compartilhar recursos de hardware gráfico entre as janelas.
Gerencia o hardware gráfico da máquina e fornece recursos para os outros softwares gráficos

Gerenciador de Janelas - É um software que controla o posicionamento e aparência de janelas dentro de um sistema de janelas em uma GUI. A maioria dos gerenciadores de janelas é projetado para fornecer um ambiente de desktop, trabalhando junto com o sistema de janelas.

Ambiente de Desktop - Trata-se de uma implementação de vários componentes, incluindo Gerenciador de Janelas, temas, bibliotecas e aplicações que interagem com o sistema de janelas presente no computador. Ex: CDE, KDE, GNOME, Xfce, LXDE

Gerenciador de sessão (Display Manager ou Login Manager) - Software que fornece uma tela para que o usuário entre com suas credenciais para login local ou remoto. Também permite escolher qual ambiente de desktop será carregado (Gnome, KDE, XFCE, etc). Ex: LightDM, gdm, xdm, kdm

Ver qual gerenciador de sessão: cat /etc/X11/default-display-manager
Ver qual gerenciador de janelas - wmctrl -m

106.3 Acessibilidade

Teclas de aderência (sticky keys) permite ao usuário digitar atalhos de teclado uma tecla por vez

Teclas de repercussão (bounce keys) serve para inibir pressionamentos de tecla não intencionais adicionando um tempo de latência entre eles, ou seja, um novo

pressionamento de tecla será aceito somente após um período de tempo especificado desde o pressionamento anterior

Teclas lentas (slow keys) exigem que o usuário mantenha a tecla pressionada por um período de tempo especificado antes de ela ser aceita

Todos os principais ambientes de desktop, especialmente o Gnome e o KDE, fornecem diversos aplicativos integrados e de terceiros para auxiliar as pessoas com deficiência visual ou mobilidade reduzida. A lição abrange os seguintes tópicos:

- Como alterar as configurações de acessibilidade.
- Maneiras alternativas de usar o teclado e o mouse.
- Adaptações da área de trabalho para os deficientes visuais.

Os comandos e procedimentos abordados foram:

- Configurações de acessibilidade do teclado: Teclas de aderência, teclas lentas, teclas de repercussão.
- Eventos do mouse gerados artificialmente.
- Teclado virtual.
- Ajustes visuais para melhorar a legibilidade.
- Temas da área de trabalho com alto contraste/letras grandes.
- Ampliadores de tela.
- O leitor de tela Orca.

107 Tarefas administrativas
107.1 Gerenciar contas de usuários e grupos e arquivos de sistema relacionados

Liçao 1

Usuários

- **useradd** - adicionar novo usuário

Parâmetros: -c; -d; -e; -f; -g; -G; -k; -m; -M; -s; -u

- **userdel** - remover usuário

Parâmetros: -r: rm diretório pessoal do usuário e todo o seu conteúdo tbm

- **usermod** - executar operações como: renomear usuário, mudar id, bloquear/desbloquear conta, alterar shell, mudar grupo, alterar diretório inicial

Parâmetros: -c; -d; -e; -f; -g; -G; -l; -L; -s; -u; -U

- passwd - trocar senha

Grupos

- **groupadd** - criar novo grupo

Parâmetros: -g;

- **groupmod** - renomear

Parâmetros: -n; -g

- **groupdel** - deletar grupo

Diretório do esqueleto (diretório o qual se baseia para criar novo usuário) - /etc/skel
IO arquivo - **/etc/login.defs** - contém algumas diretivas que especificam parâmetros de configuração para criação de usuários grupos:

- UID_MIN e UID_MAX
- GID_MIN e GID_MAX
- CREATE_HOME
- USERGROUPS_ENAB - se o sistema deve criar um novo grupo para cada nova conta de usuário com o mesmo nome do usuário
- MAIL_DIR
- PASS_MAX_DAYS
- PASS_MIN_DAYS
- PASS_MIN_LEN

- PASS_WARN_AGE

Senhas

- **passwd** - alterar senha, root pode alterar a senha de qualquer usuário, qualquer usuário pode alterar sua própria senha

Parâmetros:
-d - apaga a senha de uma conta de usuário (desabilitando o usuário).
-e - força a conta de usuário a alterar a senha.
-i -
-w -
-l - bloqueia a conta de usuário (adiciona ! no /etc/shadow)
-u - desbloqueia a conta
-n - define o tempo de vida mínimo da senha.
-x - define o tempo de vida máximo da senha.
-S - exibe informações sobre os status da senha

Grupos podem ter senhas, com o comando **gpasswd.**
Este comando também pode ser usado para adicionar e remover usuários de um grupo

- **chage -** significa “alterar idade” (change age), usado para alterar as informações de validade da senha de um usuário

Parâmetros: -d; -E; -I, -l -m; -M; -W

Lição 2

Arquivos de configuração

/etc/passwd
/etc/group
/etc/shadow
/etc/gshadow

107.2 Automatizar tarefas de administração do sistema agendando trabalhos

Lição 1
Cron

É um processo que fica residente na memória provendo o serviço de agenda de tarefas para os usuários e o sistema. Permite que um comando, programa ou script seja agendado para um determinado dia, mês, ano e hora. O Cron é adequado para

servidores e sistemas que estão constantemente ligados, pois cada cron job é executado somente se o sistema estiver rodando no horário programado

Crontabs dos usuários: Cada linha em um crontab de usuário, armazenado no diretório /var/spool/cron, contém seis campos separados por um espaço

Exemplo de crontabs do sistema no arquivo /etc/crontab: Possui campos
.---------------- minute (0 - 59)
| .------------- hour (0 - 23)
| | .---------- day of month (1 - 31)
| | | .------- month (1 - 12) OR jan,feb,mar,apr ...
| | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * * usuário command to be executed

É possível especificar vários valores usando:

*** (asterisco) -**Refere-se a qualquer valor.
, (vírgula) - Especifica uma lista de valores possíveis.
- (hífen) - Especifica um intervalo de valores possíveis.
/ (barra) - Especifica valores escalonados.

Especificações de tempo particulares
@reboot
@hourly
@daily
@weekly
@monthly
@yearly

Comandos
crontab -l - lista o que estiver agendado
crontab -e - editar seu próprio arquivo crontab
crontab -r - remove o crontab atual
crontab -u - especifica o nome do usuário cujo crontab precisa ser modificado. (root)

Diretórios
/var/spool/cron/ - diretório que armazena os arquivos de agendamentos
/etc/cron.d
/etc/cron.daily
/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly

/etc/cron.allow - usuários que estão no arquivo possuem a permissão de fazer agendamentos
/etc/cron.deny

Variáveis que podem ser alteradas no crontab
HOME, MAILTO, PATH, SHELL
cron invoca os comandos do diretório inicial do usuário, a menos que outro local seja especificado pela variável de ambiente HOME dentro do arquivo crontab. Por esta razão, podemos usar o caminho relativo do arquivo de saída e executar o script com ./foobar.sh

Lição 2
anacron e **at**

anacron - programa que executa comandos periódicos agendados, mas sem assumir que o sistema esteja continuamente ativo. Pode ser utilizado para controlar a execução de trabalhos diários, semanais e mensais em sistemas que não funcionam 24h.

/etc/anacrontab

at - agendador de tarefas semelhante ao cron, mas feito para disparar tarefas que rodam somente uma vez. Também executa tarefas mesmo que tenham passado de seu horário de execução.

O comando at permite que um horário seja informado como 1600 ou 16:00. O at também permite o uso de palavras como now (agora), noon (meio-dia), midnight (meia-noite) e teatime (16:00)

Estrutura: at 00:00 today <enter>
at> comandos
at> ctrl d

Comandos:
at - agendar jobs

- -c - Imprime os comandos de um ID de trabalho específico na saída padrão.
- -d / -r - É um alias para **atrm**.
- -f - Lê o job em um arquivo em vez da entrada padrão.
- -l - É um alias para atq.
- -m - Envia um email para o usuário no final do trabalho, mesmo se não houver saída.
- -q - Especifica uma fila na forma de uma única letra de a a z e de A a Z (por padrão, a para at e b para batch). Os jobs nas filas com as letras mais altas

são executados com um valor nice maior. Os jobs enviados a uma fila com uma letra maiúscula são tratados como trabalhos em lote (batch).
- -v - Mostra a hora em que o trabalho será executado antes de ler o trabalho.

atq - ver jobs agendados
atrm - remover jobs agendados
/etc/at.allow - semelhante ao /etc/cron.allow, se existir e tiver vazio somente o root pode fazer os agendamentos
/etc/at.deny -

Para agendar uma data em particular para executar o job, é preciso adicionar as informações de data após a hora usando um dos formatos a seguir: nome-do-mês dia-do-mês, nome-do-mês dia-do-mês ano, **MMDDYY**, **MM/DD/YY**, **DD.MM.YY** e **YYYY-MM-DD**).

System timers
Realizar alguma tarefa depois de determinado tempo de boot do sistema. **On boot sec**

Determinar um horário específico para rodar uma certa tarefa

Arquivo de configuração: /etc/systemd/system/algumacoisa.timer
Precisar ter o arquivo com o mesmo nome .service
Sintaxe: DayOfWeek Year-Month-Day Hour:Minute:Second
Comando:
systemctl list-timers
systemd-run

107.3 Localização e internacionalização

Comando **date**

date

date +“ ”

%d - dia do mês

%m - mês

%Y - ano

%H - hora

%M - minutos

%z - timezone

%T - Mostra hh:min:seg

Exemplo: date +”%T %d/%m/%Y

Arquivo que contém informações sobre todos os time zone - **/usr/share/zoneinfo/**
Arquivo que contém o time zone da máquina - **/etc/localtime**
Arquivo que contém o fuso horário padrão do sistema - **/etc/timezone**
Comandos:
timedatectl - também mostra qual é o time zone
timedatectl list-timezones - lista todos time zone

3 Formas de mudar o time zone

- Criar um link simbólico do arquivo em /usr/share/zoneinfo para o arquivo /etc/localtime
- **timedatectl set-timezone** *TIME ZONE* - muda time zone para o indicado
- **tzselect** - permite escolher o time zone

Padrões de caracteres
ASCII - Código Padrão Americano para o Intercâmbio de Informação - 7 bits
ISO-8859 - 8 bits
ISO - 8859-1
UNICODE (UTF)
UTF-8

Localização
Comando
Variável de ambiente **LANG -** ver qual a localidade
As configurações de localidade de todo o sistema são definidas no arquivo /etc/locale.conf
locale - ver variáveis relacionadas à localização do idioma e padrão de caracteres adotado.
Ex: pt_BR; en_US
locale -a - mostra as localização disponíveis
localectl set-locale LANG=en_US.UTF-8 - altera localidade do sistema
iconv - converte texto de um formato de caractere para outro formato