Curso de Formação LPIC-1

Exame 101




Bruno Santos <brunosantos@whatgeek.com.pt>

Curso Linux: formação

- Linux boot process
  - Boot loaders
  - Processo de arranque de Linux
  - Runlevels e init
  - Desligando o sistema
  - Processo init
  - /etc/inittab
  - Ctrl+Alt+Delete
  - Directorias de runlevel
  - Configurar runlevels
  - Single user mode

Linux boot process

Boot loaders

- Um sistema Linux tem o *boot loader* num de dois lugares
  - O m*aster boot record (MBR)*
  - Primeiro sector da partição de Linux
- Como outros sistemas operativos usam a mesma metedologia – particularmente Windows, reclamando a mesma localização do MBR que o Linux – podem ocorrer problemas
- Fazer um *dual boot* com Linux e Windows pode ser um desafio
- Instalados na ordem correcta, usando LILO ou GRUB para gerir o ambiente do *boot loader,* torna-se muito simples
- É recomendado instalar-se o Windows primeiro, devido a ser mais esquisito com o MBR
- Após o ultimo *reboot,* instalar Linux no espaço livre, permitindo ao LILO ou GRUB aperceberem-se e configurarem a partição Windows no setup do *boot loader*

Linux boot process

Boot loaders: LILO – The Linux Loader

- O pacote do LILO consiste
  - O comando /sbin/lilo
  - Ficheiro de configuração /etc/lilo.conf
  - Documentação de ajuda
  - Ficheiros do *boot loader*
- Relação entre o comando /sbin/lilo, os ficheiros de configuração, e o MBR/primeiro sector
  - O programa /sbin/lilo é invocado
  - O ficheiro /etc/lilo.conf é lido
  - Os ficheiros mencionados no ficheiro são localizados. São produzidos erros se não forem encontrados
  - A imagem binária resultante é escrita por defeito para o MBR
  - O programa /sbin/lilo termina

Linux boot process

Boot loaders: LILO – The Linux Loader

- O ficheiro de configuração /etc/lilo.conf é divido em duas partes
  - A primera secção do topo é a secção global
    - Contém definições que afectam todas as imagens e o comportamento do ambiente do LILO
  - A segunda secção é a das imagens
    - Uma imagem é apenas um conjunto de parametros de configuração que constituem uma referência de boot e opções para a selecção das opções que aparecem no menu do LILO

Linux boot process

Boot loaders: LILO – The Linux Loader

```
prompt
timeout=50
default=win
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32

image=/boot/vmlinuz-2.6.29-10
        label=win
        initrd=/boot/initrd-2.6.29-10.img
        read-only
        root=/dev/hda5

other=/dev/hda1
        optional
        label=win
```

Linux boot process

Boot loaders: LILO – The Linux Loader

- Alguns parametros do LILO
  - timeout – isto é 1/10 de segundo
  - default – o valor para o parametro por defeito. TEM QUE CORRESPONDER um label de uma imagem
  - *lba32* vs *linear* – se usamos um disco grande, ou a partições (/boot e /) estão acima do cilindro 1024, lb32 é recomendado. SCSI ou outros tipos de discos, *linear* é recomendado

Linux boot process

Boot loaders: Grand Unified Boot Loader - GRUB

- GRUB é mais recente que o LILO
- Oferece funcionalidades superiores
  - Poder editar o menu antes do arranque
- O ficheiro de configuração do grub é */boot/grub/menu.lst*
- Contém vários items, incluindo alguns semelhantes ao LILO

```
default 0
timeout 10
splashimage=(hd0,4)/boot/grub/splash.xpm.gz

title Gentoo Linux 2.6.30-gentoo-r4
root (hd0,4)
kernel /boot/kernel-2.6.30-gentoo-r4 root=/dev/sda3
video=uvesafb:mtrr:3,ywrap,1680x1050-32@60

title
rootnoverify (hd0,0)
makeactive
chainloader +1
```

Linux boot process

Boot loaders: Como o grub trata os discos

- O GRUB não é capaz de diferenciar entre discos IDE, SCSI
- Apenas lê as entradas da BIOS e parte desde esse ponto
- Para tornar as coisas mais previsiveis, detecta e usa discos IDE antes dos discos SCSI
- A numeração dos discos e partições é um pouco diferente dos outros *boot loaders*
- Primeiro disco e primeira partição do mesmo – *(hd0,0)*
- O grub numera os discos a partir do 0, com 0-3 sendo usualmente discos IDE
- A partições são numeradas da mesma forma, com 0 a 3 sendo as primárias e 4 sendo sempre a primeira partição lógica no disco
- Designação da primeira drive de disquetes do sistema - (fd0)

Linux boot process

Boot loaders: configurar o GRUB

- A instalação do GRUB consiste em
  - Correr o comando *grub-install* com parametros
  - Executar *grub* e configurar o *boot loader* enquanto dentro do GRUB

```
#Instalar no primeiro disco (consequentemente no MBR do disco)
grub-install /dev/hda
```

- Instalar o GRUB na directoria /boot
- Torna-se necessário num sistema com uma partição de /boot pequena para conter apenas os ficheiros de boot e evitar o problema dos primeiros 1024 cilindros

```
#Instalar na directoria /boot
grub-install -root-directory=/boot  /dev/hda
```

Linux boot process

Boot loaders: configurar o GRUB

- Não é necessário usar apenas o script *grub-install*
- É possivel configurar o GRUB desde a prompt – grub>
- Por outra palavras, é possivel usar-se outro *boot loader* e decidir-se usar o GRUB sem se conhecer alguns detalhes, como por exemplo a localização dos ficheiros de estágio do GRUB

Linux boot process

Boot loaders: Instalar nativamente o GRUB

```
grub
#Se não soubermos que dispositivo contém os ficheiros de estágio do grub
#digitamos grub ( e <TAB>
grub> root (
#se formos apresentados com vários dispositivos e partições, procuramos pelo ficheiro loader stage1
#O resto do grub deverá estar junto
grub> find /boot/grub/stage1
#o comando procura pelo stage1 e mostra em que dispositivo está. Podems depois usar esse
#dispositivo
grub> root (hd0,1)
grub> setup (hd0)
#Este comando configura o GRUB no MBR do dispositivo especificado
#Se necessario configurar para o sector de boot de uma particao
grub> setup (hd0,1)
grub> boot
```

- Da proxima vez que inicializarmos o sistema, o menu do GRUB deverá aparecer e arrancar com a opção por defeito sem qualque acção por nossa parte

Linux boot process

Boot loaders: Instalar nativamente o GRUB

- Se o GRUB não estiver configurado com imagens e apenas aparecer a prompt grub>, é possivel na mesma efectuar um *boot* ao sistema

```
grub> root (hd0,1)
grub> kernel /boot/vmlinux-2.6.29-10
```

- Se for necessário, podem-se carregar módulos, caso algum dispositivo seja necessário para carregar o resto do sistema

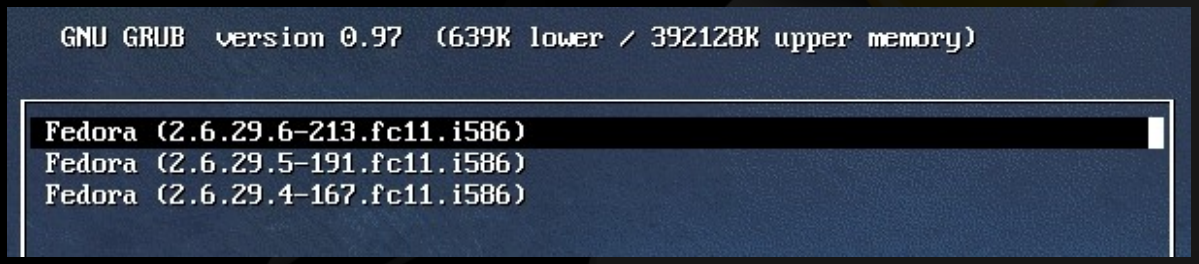
```
grub> module /boot/algumodulo
grub> boot
```

Linux boot process

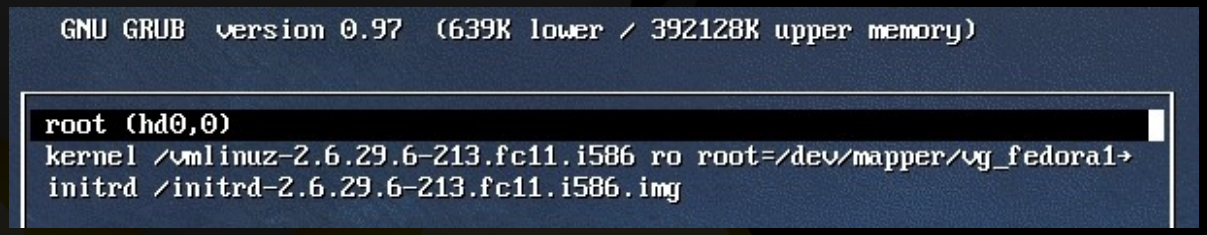
Boot loaders: Editar linhas do GRUB

- Outra das funcionalidades do GRUB é a possibilidade de editar as linhas de arranque do sistema e adicionar novas opções para o arranque

Menu do GRUB, pressionar <ESC> para parar a contagem
Escolher a linha que queremos editar e pressionar a tecla <e> (editar)



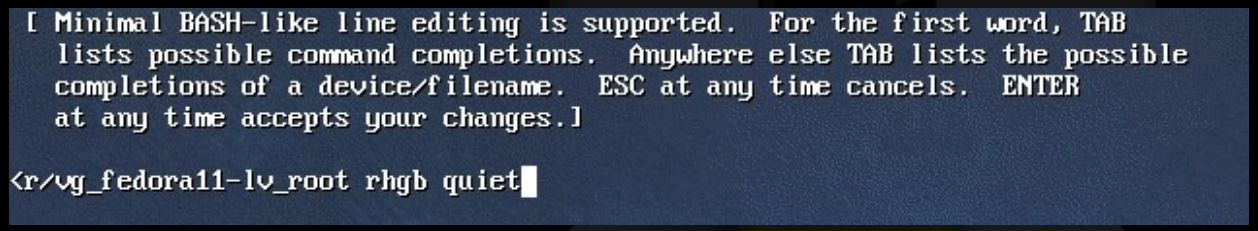
Irão aparecer as linhas correspondentes à entrada que seleccionamos
Escolher novamente a linha a editar e pressionar <e> (editar)



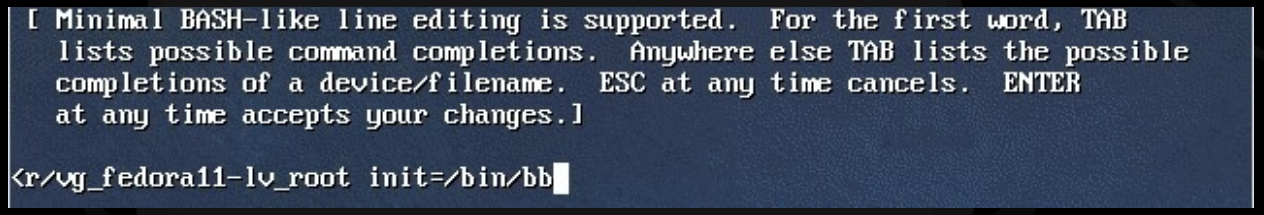
Linux boot process

Boot loaders: Editar linhas do GRUB

Iremos ser levados para o final da linha.
A partir daqui, podemos editar o que desejarmos, e que seja reconhecido pelo kernel



Por exemplo, ao acrescentar a linha init=/bin/bb estamos a informar o kernel que assim que terminar a sua execução, e passe para o init, que execute a aplicação bb



Assim que terminar-mos de editar a linha, pressionamos <enter> e posteriormente <b> para o GRUB arrancar

Linux boot process

Visualizar mensagens do arranque

- Visualizar as mensagens do arranque, porque aconteceu algum erro ou apenas por curiosidade é efectuado com o comando *dmesg*
- A maioria pensa que o comando *dmesg* mostra todas as mensagens ou tudo desde a primeira mensagem até ao ecrã de login
- Não é verdade
- O comando *dmesg* apenas mostra as mensagens que ocorrem antes do sistema entrar no *runlevel* por defeito e as mensagens que terminam em [OK] ou [Failed] começam a aparecer
- As distribuições mais recentes podem não mostrar mensagens mas sim um login gráfico e um ecrã de arranque
- Isto pode ser desactivado no LILO ou no GRUB
- As mensagens de arranque são guardadas no ficheiro var/log/dmesg e podem ser visualizadas num longo ecrã correndo o comando
- Para mostrar as mensagens de uma forma mais amigável, efectuamos o PIPE do comando
  - dmesg | more
  - dmesg | less

Linux boot process

O processo de arranque de Linux

- Os passos de arranque de Linux são semelhantes por várias distribuições
  - Excepto slackware e variantes de BSD
- Passos
  - Ligar o computador
  - O código da BIOS (basic input/output system) carrega e procura um sector de arranque (boot) válido
  - A BIOS carrega o sector de arranque de uma disquete, disco ou CD-ROM, de acordo com a ordem de procura
  - O Master Boot Record (MBR) é lido. Está nos primeiros 512bytes do primeiro sector do primeiro disco activo
  - Qualquer código de arranque encontrado é executado
  - Se usando o LILO (LInux LOader), o código de menu (geralmente em */boot/boot.b*) é mostrado.A prompt do LILO é geralmente LILO: em modo de texto ou um menu gráfico com várias opções
  - Se usando o GRUB (GRand Unified Bootloader), o código do menu é mostrado (geralmente em */boot/grub/[grub.conf|menu.lst])*

Linux boot process

O processo de arranque de Linux

- (continuação)
  - O utilizador escolhe ou digita uma opção ou o tempo de escolha termina e a opção por defeito é seleccionada
  - Se escolhemos uma imagem de Linux, o caminho para o kernel é lido e a imagem comprimida de Linux é executada
  - Enquanto o kernel está a carregar, ele inicializa os dispositivos, carrega módulos necessários, procura pelo *initial* RAM disk (initrd) e carrega-o se necessário. Monta depois o sistema de ficheiros principal, como definido pelo parametro *root=/dev/xxx*
  - O programa */sbin/init* é carregado e torna-se no PID 1, o avô de todos os restantes processos no sistema
  - O processo *init* lê o ficheiro */etc/inittab* e executa os scripts em /etc/init.d/rcS para Debian e /etc/rc.d/rc.sysinit para Red Hat

Linux boot process

O processo de arranque de Linux

- (continuação)
  - O script de inicialização carrega todos os módulos necessários, verifica o sistema de ficheiros *root,* monta os sistemas de ficheiros locais, inicializa os dispositivos de rede e monta os sistemas de ficheiros remotos (se configurados)
  - O processo *init* lê novamente o ficheiro */etc/inittab* e altera o sistema para o *runlevel* por defeito, executando os scripts no directorio apropriado
  - Os scripts do *runlevel* são executados, com os scripts *SXXservice* na directoria por do *runlevel* por defeito a serem executados por ordem numérica
  - As sessões *mingetty* especificadas em */etc/inittab* são carregadas e o sistema (e login) mostra a prompt e espera pelo utilizador para iniciar a sua sessão

Linux boot process

Runlevels e *init*

- Um *runlevel* é uma definição do estado do sistema, onde o número (por definição) representa um conjunto de serviços e algumas aplicações que são executadas
- Todas as distribuições suportam a presença dos *runlevels* 0 a 6
- Existem nove *runlevels* no total, mas nenhuma distribuição de Linux usa mais que seis
- Para algo que existe em todos os Linux, podia-se pensar que eram standard, mas não são
- A maioria das distribuições usa o seguinte esquema
  - 0 – Desliga o sistema.
  - 1 – Single user/Modo de manutenção. Não existem utilizadores na maquina, apenas o sistema, uma consola e algumas ferramentas
  - 2 – Indefinido por defeito. Em sistemas Red Hat é configurável
  - 3 – Multi-utilizador. Capacidade completa de rede. Login de texto
  - 4 - Indefinido por defeito. Em sistemas Red Hat é configurável

Linux boot process

Runlevels e *init*

- (continuação)
  - 5 – Multi-utilizador. Capacidade completa de rede, modo gráfico. X corre por defeito
- Num sistema a correr, para saber qual o *runlevel* em que estamos, executamos o comando *runlevel*

```
runlevel
N 3
```

- O primeiro caracter representa o *runlevel* anterior
- Um N indica que o sistema arrancou e foi para o *runlevel* por defeito, indicado pelo segundo caracter
- Se o sistema estivesse anteriormente num *runlevel*, o primeiro caracter reflectia esse *runlevel*
- Modificar o *runlevel* do sistema numa consola, envolve o comando *init* ou *telinit*, que é um *link* para o comando *init*. Existe apenas para combatibilidade

Linux boot process

Runlevels e *init*

- Para levar o sistema para o *runlevel* 3

```
init 3
```

- O sistema mostra mensagens consistentes com a saída do *runlevel* corrente e entrando no novo *runlevel*
- Levar o sistema para o *runlevel* 5

```
init 5
```

- Se desejamos manter o *uptime* do sistema, mantendo-o em execução, não é preciso efectuar *reboot*
- Leva-se o sistema para o *runlevel 1* e novamente para um *runlevel* mais alto
- Ao efectuar-mos isto, a maioria das mensagens de erro são corrigidas e torna o sistema limpo

```
init 1; init 3
```

Linux boot process

Desligando o sistema

- Para se desligar um sistema, pode-se usar o comando *init* ou o comando *shutdown*

```
shutdown -t # -options time message
```

- Opções para o comando incluem
  - -h : desliga o sistema quando desligar todos os processos (*runlevel 0*)
  - -r : faz *reboot* ao sistema
  - -t# : Onde # é o tempo em segundos para esperar antes de começar a enviar sinais aos processos
  - -f : não força uma verificação (*fsck*) ao reiniciar
  - -F : força uma verificação (*fsck*) ao reiniciar
  - -c : cancela um shutdown em progresso
  - -a : usa o ficheiro */etc/shutdown.allow*

Linux boot process

Desligando o sistema

- Utilizar o comando *shutdown* pode ser simples ou complexo

```
shutdown now
```

- Este comando desliga o sistema imediatamente, enviando o sinal SIGTERM a todos os processos
- Envoca posteriormente o processo *init* para levar o sistema ao *runlevel* 1

- Desliga o computador completamente

```
shutdown -h now
```

- Reinicia o computador após desligar

```
shutdown -r now
```

Linux boot process

Desligando o sistema

- Avisar todos os utilizadores que o sistema vai desligar às 17:00, com uma mensagem

```
shutdown -h 17:00 “O sistema vai desligar às 17:00”
```

- O tempo pode ser especificado
  - Em absoluto: 17:30
  - Número de minutos no futuro: +10
- Usar o *shutdown* com um atraso, causa o sistema criar o ficheiro */etc/nologin*, que desactiva o login de utilizadores
- O ficheiro é removido antes das alterações ao *runlevel*

Linux boot process

Desligando o sistema

- O ficheiro */etc/shutdown.allow* é desenhado para conter *usernames* para permitir que estes possam desligar o sistema sem previlégios de *root*
- Este ficheiro permite um máximo de 32 utilizadores
- É apenas lido quando invocado com a opção -a do comando *shutdown*
- Outros comandos para parar ou desligar o sistema incluem
  - halt
  - poweroff
  - reboot
- Estes ultimos são *links* para o primeiro

Linux boot process

Desligando o sistema

- O comando *halt* é directamente usado nos *runlevels* 0 e 6
- Outros *runlevels* chamam o comando *shutdown* com a opção de reiniciar ou parar o sistema
- Este comando pode ser ultrapassado com o uso do comando *halt -f* para forçar um sistema a parar
- O comando *reboot* é um *link* para o comando *halt*, que chama o comando *shutdown* com o parametro -r para efectuar um *reboot* ao sistema

Linux boot process

O processo *init*

- Quando executado, o programa */sbin/init* torna-se no primeiro ou processo parente e é lhe atribuído o PID 1
- Todos os restantes processos são filhos do *init* e herdam o seu ambiente e atributos por defeito

Linux boot process

O ficheiro */etc/inittab*

- O sistema, e em particular o processo *init* referem a este ficheiro
- É responśavel por
  - *runlevels*
  - Quantas funções de programas respondem a estimulos
  - Como o sistema é inicializado
- Entradas no ficheiro */etc/inittab* consistem
  - id:runlevels:acção:processo

- Os terminais virtuais (tty1 até tty6 por defeito), os processos agetty são configurados com as seguintes linhas

```
# TERMINALS
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:2345:respawn:/sbin/agetty 38400 tty2 linux
c3:2345:respawn:/sbin/agetty 38400 tty3 linux
c4:2345:respawn:/sbin/agetty 38400 tty4 linux
c5:2345:respawn:/sbin/agetty 38400 tty5 linux
c6:2345:respawn:/sbin/agetty 38400 tty6 linux
```

Linux boot process

O ficheiro */etc/inittab*

```
c6:2345:respawn:/sbin/agetty 38400 tty6 linux
```

- A primeira coluna é o ID
  - Neste caso, é a string associada com o tty a ser configurado para as contas do sistema. Pode ser qualquer outro número
- A segunda coluna são os *runlevels* onde este comando ou serviço é permitido correr ou onde o programa deve ter acção
- A terceira coluna é a acção que deve ser tomada para esta entrada
- A quarta coluna é o comando ou serviço a ser afectado ou a correr, bem como as opções e argumentos precisos

Linux boot process

O ficheiro */etc/inittab*

```
c6:2345:respawn:/sbin/agetty 38400 tty6 linux
```

- Valores importantes para a terceira coluna (coluna de acção)
  - respawn : Reinicia o processo quando este for terminado
    - É usado principalmente para o X e terminais de login (mingetty, getty e agetty)
  - sysinit : é executado antes de entrar no *runlevel* por defeito
  - initdefault : define o *runlevel* por defeito
  - ondemand : entradas com esta acção podem ser invocadas com o comando *init* e opções a, b ou c
    - Não existem alterações no *runlevel,* apenas a invocação do programa configurado
  - once : isto causa a invocação das entradas apenas uma vez, quando entrado no *runlevel* especificado
  - wait : esta acção é para processos que têm que ser terminados antes de qualquer coisa acontecer
    - Geralmente inicialmente, no inicio da entrada no *runlevel*
    - Posteriormente, o *init* continua a sua execução

Linux boot process

O ficheiro */etc/inittab*

- O *runlevel* por defeito do sistema permite ao sistema saber qual o *runlevel* para onde desejamos ir após todos os scripts inicialização *(sysinit)* forem terminados
- Esta entrada é importante porque o sistema vai para este *runlevel* sempre que arranca ou é reinicializado

```
id:3:initdefault:
```

- A entrada da inicialização do sistema coloca o script ou programa que é executado após a inicialização do sistema e antes do *runlevel* por defeito ser alcançado
- Não existem *runlevels* para esta entrada, eles ocorrem posteriormente

```
# System initialization, mount local filesystems, etc.
si::sysinit:/sbin/rc sysinit
```

Linux boot process

Agarrando Ctrl+Alt+Delete

- Existe uma determinada entrada no ficheiro */etc/inittab* que executa uma determinada acção por defeito quando as teclas Ctrl+Alt+Delete são pressionadas, normalmente para iniciar um *shutdown -h*
- É possivel alterar esta linha para tornar impossivel alguém que pressione essas teclas
- Numa maquina *Red Hat,* o ficheiro */etc/inittab* tem a seguinte linha para configurar o que aconteçe com Ctrl+Alt+Delete

```
ca:12345:ctrlaltdel:/sbin/shutdown -t3 -r now
```

- Esta linha espera que o Ctrl+Alt+Delete seja pressionado, inicia um *shutdown* e efectua um *reboot* com três segundos de atraso antes de enviar sinais para os processos
- No entanto, inicia o *shutdown* imediatamente
- Pode ser catrastrófico num sistema em produção. Para alterar essa situação:

```
ca:12345:ctrlaltdel:echo “Não há Ctrl+Alt+Delete”
```

Linux boot process

Agarrando Ctrl+Alt+Delete

- O processo *init* não lê o ficheiro */etc/inittab* excepto se o sistema for reiniciado ou lhe for dito para o fazer
- Se foram efectuadas alterações ao ficheiro */etc/inittab*, o comando seguinte indica ao processo *init* para re-ler o ficheiro

```
init q
```

Linux boot process

As directorias dos *runlevels*

- As duas maiores variações de startup de sistema são
  - Método BSD/Slackware
    - Poucos, mas mais longos *scripts* de inicialização
  - Método SysV
    - Envolve o script *sysinit* e um determinado número de directorias que contêm *links* para os serviços do sistema
    - Cada directoria representa um *runlevel*
- Red Hat e Debian usam uma estrutura semelhante, mas com uma grande diferença

Red Hat

/etc

- init.d
- rc.d
  - rc0.d
  - rc1.d
  - rc2.d
  - rc3.d
  - rc4.d
  - rc5.d
  - rc6.d
  - rc.sysinit

Debian

/etc

- init.d
- rcS.d
- rc0.d
- rc1.d
- rc2.d
- rc3.d
- rc4.d
- rc5.d
- rc6.d

Linux boot process

As directorias dos *runlevels*

- A directoria *init.d* em cada caso contém todos os serviços configurados executáveis
- Estes estão *linkados* com as directorias dos *runlevels* por dois tipos de scripts
  - Scripts de arranque (start scripts)
  - Scripts de finalização (kill scripts)
- Ambos os scripts são *links* para os serviços executáveis na directoria *init.d*
- Os scripts de arranque são semelhantes a ***SXXservico***
  - O S maiusculo indica ao servico para arrancar
  - O XX é um valor numérico que forçam a ordem de arranque
  - O serviço é uma mneónica para o nome do serviço
- Scripts de finalização usam a mesma convenção, com a excepção de usarem um K em vez de um S
- A presença do K indica ao serviço para se terminar graciosamente
- Após a inicialização do sistema estar terminada, o processo *init* lê o ficheiro */etc/inittab* e procura a linha que indica o *runlevel* por defeito

Linux boot process

As directorias dos *runlevels*

- Após o valor ser determinado, o processo *init* procura a directoria do *runlevel* por defeito
- Após entrar na directoria, inicia os scripts de inicialização por ordem do valor XX
- Quando o *runlevel* é alterado, o sistema toma nota dos processos no *runlevel* em execução, inspecciona os scripts do *runlevel* para onde vai, e deixa os que estão com um S no *link* em execução
- Se não existe S no *link* do serviço no *runlevel* pretendido, é considerado *undefined* e o script com K no *link* para o serviço é executado antes do *runlevel* ser alterado

Linux boot process

Configurar os *runlevels*

- Existem vários métodos de alterar ou configurar os scripts de arranque ou finalização dos vários *runlevels*
  - Manualmente – Isto envolve o uso dos comandos *rm*, *ln* e *mv* para manipular os *links* que apontam para os serviços na directoria *init.d*
  - *Scriptable* – isto envolve usar os scripts *chkconfig*, *update-rc.d* e outras ferramentas não-interactivas
  - GUI – Envolve usar o *ntsysv*, *SysVConfig*, *tksysv* e outras ferramentas gráficas
- Red Hat inclue tanto o *ntsysv* e o *chkconfig*
- Debian usa uma ferramenta chamada *update-rc.d*

Linux boot process

Configurar os *runlevels*

- Usar a ferramenta *ntsysv* para configurar os serviços é fácil e rápido
- Por defeito edita o *runlevel* corrente
- Após invocar *ntsysv*, aparece uma lista de serviços que podem ser ligados ou desligados
- Basta pressionar a barra de espaços para ligar ou desligar o serviço
- É preciso sair do *runlevel* corrente para as alterações terem efeito

Para configurar outro runlevel
ntsysv --level 5

Para configurar vários runlevels ao mesmo tempo
Ntsysv --level 016

Linux boot process

Configurar os *runlevels*

Linux boot process

Configurar os *runlevels*

- Uma das ferramentas mais uteis para configurar *runlevels* é o *chkconfig*
- Não tem nenhum menu, não é interactivo e é um script
- Usar o *chkconfig* para listar a configuração corrente

```
chkconfig --list
NetworkManager    0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd               0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
avahi-daemon      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bluetooth         0:off   1:off   2:off   3:on    4:on    5:on    6:off
cpuspeed          0:off   1:on    2:on    3:on    4:on    5:on    6:off
firstboot         0:off   1:off   2:off   3:off   4:off   5:off   6:off
haldaemon         0:off   1:off   2:off   3:on    4:on    5:on    6:off
httpd             0:off   1:off   2:off   3:off   4:off   5:off   6:off
ip6tables         0:off   1:off   2:on    3:on    4:on    5:on    6:off
iptables          0:off   1:off   2:on    3:on    4:on    5:on    6:off
irda              0:off   1:off   2:off   3:off   4:off   5:off   6:off
irqbalance        0:off   1:off   2:off   3:on    4:on    5:on    6:off
iscsi             0:off   1:off   2:off   3:on    4:on    5:on    6:off
iscsid            0:off   1:off   2:off   3:on    4:on    5:on    6:off
mdmonitor         0:off   1:off   2:on    3:on    4:on    5:on    6:off
messagebus        0:off   1:off   2:on    3:on    4:on    5:on    6:off
multipathd        0:off   1:off   2:off   3:off   4:off   5:off   6:off
netconsole        0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

Linux boot process

Configurar os *runlevels*

- O *chkconfig* é um comando util porque mostra o estado dos serviços em cada um dos *runlevels*

Para configurar um servico para determinados *runlevels*
chkconfig --levels 345 smb on

- Este comando faz com que o serviço smb (samba) seja executado nos *runlevels* 3, 4 e 5

Pode ser confirmado com o seguinte comando
chkconfig --list | grep smb

- Se quisermos que um serviço seja gerido pelo *chkconfig*

chkconfig --add tarfoo

- Este comando faz com que o serviço *tarfoo* ter uma entrada de arranque e término em determinados *runlevels*, tornando-o completamente configurável pelo *chkconfig*

Linux boot process

Single user mode

- Às vezes, entrar em *single mode* é necessário, mesmo desejável, para um administrador
- Existe alguma confusão sobre as diferenças entre *runlevels* S, s e 1
- Os *runlevels* s e S são o mesmo e existem primeiramente para executar os scripts associados com o *runlevel* 1 antes de ser iniciada a transferência para o *runlevel 1*
- Uma situação que por vezes confronta os administradores é um sistema que não entra correctamente no *runlevel* ou no caso do *runlevel 5*, não inicia o X

- Somos um administrador de sistemas que configurámos uma máquina para ir para o *runlevel* 5 por defeito
- É suposto terminar numa sessão de X
- Se usarmos o grub, editamos a linha e acrescentamos **single** à linha

Curso Linux

bibliografia

- LPIC I, Exam Cram 2, Brunson - QUE Certification
- LPI Linux Certification In a Nutshell, Pritchard, Pessanha, Langfeldt, Stranger & Dean – O REILLY
- Linux Administration Handbook, Second edition, Nemeth Snyder Hein – Prentice Hall