## SEQUÊNCIA DE BOOT

**1 – BIOS**

Carrega setor de boot de um dos dispositivos: Floppy, CDROM ou HD.

A sequência de boot pode ser alterada no Setup da BIOS, acessada pressionando uma tecla durante o boot. A tecla pode variar, vas normalmente é Del, F1, F2, ou F10.

**2 – Master Boot Record (MBR)**

Primeiros 512bytes do HD.

**3 – Gestor de inicio (GRUB)**

O Grub é muito grande para ficar no MBR. Assim, apenas o 1º estágio fica lá.

O 1º estágio aponta para o segundo estágio, que fica no diretório /boot.

**4 – Kernel (vmlinuz + initrd)**

O grub carrega o Kernel, que usa o ramdisk inicial initrd.

**5 – init**

O init é o primeiro processo a ser executado depois que o kernel é carregado. Ele se encarrega de chamar os scripts de inicialização do Run Level escolhido.

**6 – Run Levels**

*Linux Standard Base - Core Specification 4.0*

*20.5. Run Levels*

*The following run levels are specified for use by the Default-Start and Default-Stop actions defined in Comment Conventions for Init Scripts as hints to the install_initd command. Conforming implementations are not required to provide these exact run levels or give them the meanings described here, and may map any level described here to a different level which provides the equivalent functionality. Applications may not depend on specific run-level numbers.*

*0 halt*
*1 single user mode*
*2 multiuser with no network services exported*
*3 normal/full multiuser*
*4 reserved for local use, default is normal/full multiuser*
*5 multiuser with a display manager or equivalent*
*6 reboot*

*Note: These run levels were chosen as reflecting the most frequent existing practice, and in the absence of other considerations, implementors are strongly encouraged to follow this convention to provide consistency for system administrators who need to work with multiple distributions.*

## GRUB (0.X)

Arquivo de configuração */boot/grub/menu.lst:*

```
default 0
timeout 5
fallback 1
splashimage=(hd0,0)/grub/splash.xpm.gz

title Fedora  (2.6.35.6-45.fc14.x86_64)
root (hd0,0)
kernel /boot/vmlinuz- 2.6.35.6-45.fc14.x86_64 ro root=/dev/hda1
initrd /boot/initrd-2.6.35.6-45.fc14.x86_64.img
```

Detalhando:

**default** – Indica qual sistema operacional apresentado no menu do grub será usado por padrão para dar o boot.
**timeout** – O menu irá aguardar por 5 segundos alguma tecla ser pressionada antes de inicializar o SO **default**. Qualquer tecla pressionada irá cancelar o timer e a inicialização será feita pressionando a tecla ENTER.
**fallback** – Em caso de erro ao tentar dar o boot no SO **default**, uma nova tentativa será feita com o SO indicado aqui.
**splashimage** – Indica a localização da imagem de fundo do Grub.
**title** – Define um titulo para o SO que será mostrado no menu do Grub.
**root** – Indica a partição onde o SO está localizado no HD.

*(hd0,0) - /dev/hda1*
*(hd0,1) - /dev/hda2*
*(hd1,0) - /dev/hdb1*
*(hd1,1) - /dev/hdb2*

**kernel** – Indica a localização da imagem do kernel. Alguns parâmetros podem ser informados.
**Initrd** – Indica a localização do ramdisk inicial que será usado.

## GRUB2 (1.X)

Novidades do Grub2 em relação ao Grub:

- Nenhum menu será mostrado caso o computador não possua outro sistema operacional.
- O arquivo */boot/grub/menu.lst* foi trocado por */boot/grub/grub.cfg*.
- O arquivo grub.cfg será sobrescrito em caso de update, adição ou remoção de kernel ou se o usuário executar um *update-grub2*.
- O usuário pode editar o arquivo */etc/grub/40_custom*, onde ele pode colocar suas próprias entradas do grub. Esse arquivo não será sobrescrito.
- O arquivo principal de configurações do Grub é o */etc/default/grub*.
- Nenhuma alteração de configuração fará efeito até que seja executado o comando *update-grub2*.

Opções do arquivo */etc/default/grub*:

- GRUB_DEFAULT
  - Seleciona o item default no menu do grub. O primeiro é 0, o segundo é 1 e assim por diante.
- GRUB_TIMEOUT
  - Seleciona quanto tempo em segundos o menu será exibido antes de inicializar pela opção padrão.
- GRUB_HIDDEN_TIMEOUT (Em computadores que tenham apenas um sistema operacional)
  - 0 – O menu não será mostrado. O sistema irá inicializar imediatamente.
  - X, sendo X > 0 – O sistema sirá esperar X segundos exibindo uma tela limpa antes de dar o boot. nenhum menu será mostrado.
  - Vazio – O menu será mostrado pelo tempo definido em GRUB_TIMEOUT.
- GRUB_HIDDEN_TIMEOUT_QUIET
  - true – Nenhum contador será mostrado. A tela ficará vazia.
  - false – Um contador será exibido em uma tela limpa pelo durante o tepo do GRUB_HIDDEN_TIMEOUT

## SYSV INIT / UPSTART

**SysV init**

O SysV init tem sido o inicializador do sistema padrão no Linux desde o início de sua fase adulta. Ele foi introduzido pelo Unix System V, da AT&T, a partir de 1983 e seu modelo serviu de inspiração para o Linux. O modelo estabelecido pelo SysV init só é completo com o uso dos Run Levels.

Quando o sistema é inicializado, depois que o kernel termina de carregar, o init é chamado recebendo como parâmetro o runlevel que ele deve executar. De acordo com o runlevel, o SysV init inicia os scripts correspondentes ao nível de execução desejado.

Para cada Run Level temos um diretório no sistema que contém os scripts (ou links para eles) que devem ser inicializados ou parados. Por exemplo, em um servidor RHEL5, podemos achar os seguintes scripts no diretório correspondente ao Run Level 3:

*root@server:~# ls /etc/rc3.d/*

| | | | |
|---|---|---|---|
| *S10rsyslog* | *S16ssh* | *S21fam* | *S99rc.local* |
| *S12acpid* | *S20cprint* | *S24hal* | *S99rmnologin* |
| *S12dbus* | *S20cups* | *S89cron* | *S99stop-bootlogd* |

Perceba que no nome do script há ainda um **S**, indicando que o script receberá *start* como parâmetro (seria K, de Kill, para o script receber o parâmetro *stop*), e um número que indica a ordem que ele deve ser executado. A inicialização é feita sequencialmente, sem paralelismo.

Nem todas as distribuições seguem a risca o padrão de Run Levels. No caso do Debian e derivados, os níveis de 2 a 5 são idênticos. Nesse caso, a interface gráfica é sempre iniciada caso esteja instalada.

Para que um serviço seja gerenciado pelo SysV init, o script de inicialização deve ficar dentro do diretório */etc/init.d/* e os links simbólicos correspondentes devem ser criados nos diretórios */etc/rcX.d*, onde "X" é o RunLevel, com o nome precedido da ação (K ou S) e da prioridade (ordem de precedência) que ele será executado.

A maioria das distribuições possuem aplicativos que facilitam a criação desses links simbólicos. O Debian e suas derivações (incluindo o Ubuntu) oferecem o comando *update-rc.d*. Depois de criado o script em /etc/init.d. execute:

*update-rc.d script start 20 2 3 4 5 .*
*update-rc.d script defaults*

O controle dos serviços deve ser feito pelos comandos a seguir:

*/etc/init.d/meuservico start*
*/etc/init.d/meuservico stop*
*/etc/init.d/meuservico status*

Os parâmetros passados devem ser tratados pelo script *meuserviço*. No caso do Red Hat e derivados (Fedora, CentOS, …) existe o *chkconfig.* Assim, um equivalente no Red Hat para o comando *update-rc.d* anterior seria:

*chkconfig --level 345 meuservico on*

Um detalhe importante no *chkconfig* é que ele exige no mínimo duas linhas de comentários no script que será gerenciado:

```
# chkconfig: 345 80 20
# description: The Apache HTTP Server is an efficient and extensible  \
#              server implementing the current HTTP standards.
```

A primeira linha indica que o script receberá o parâmetro *start* nos runlevels 3, 4 e 5, com prioridade 80 e *stop* nos demais runlevels, com prioridade 20.

A segunda linha é apenas uma descrição, mas é obrigatória. Essa descrição é apresentada quando usamos o comando *ntsysv* (em desuso – um equivalente ao *chkconfig* que usa dialog), pressionando *F1* no serviço desejado.

**Upstart**

Definição da Canonical: *"Upstart é um substituto do daemon /sbin/init baseado em eventos que inicia serviços durante o boot, para-os durante o desligamento e supervisiona-os enquanto o sistema está em funcionamento."*

O Upstart resolve 2 problemas existentes no modo de funcionamento do SysV init. O primeiro é que o Upstart executa os scripts de inicialização de forma paralela, onde o script depende apenas que os eventos estabelecidos como pré-requisitos ocorram. Isso reduz drasticamente o tempo de boot. O segundo é que o Upstart tem a capacidade de monitorar um serviço e, em caso de interrupção abrupta, inicializá-lo novamente. O fato de os eventos determinarem quando um serviço é inicializado ou parado traz vantagens para o sistema, já que os eventos podem ser gerados por qualquer processo.

Como forma de manter a compatibilidade com o SysV init, propiciando uma migração suave, existe um script de Upstart que chama o SysV init(*/etc/rc.sysinit*) passando o run level desejado.

Os scripts upstart ficam no diretório é o */etc/init* (em versões anteriores, os scripts ficavam em */etc/event.d*).

Em ambos os casos, os scripts tem a extensão *.conf* e possuem três sessões principais:

- Condições para inicializar ou parar o serviço (evento).
  Ex.:
  *stop on runlevel [016]*
- Ações a serem tomadas antes de inicializar e depois de parar o serviço.
  Ex.:
  *pre-start script*
  *mkdir /tmp/meuservico*
  *end script*
- Comando ou script que inicializa o serviço.
  Ex.:
  *exec /usr/bin/meuservico.sh*

O controle dos serviços deve ser feito pelos comandos a seguir:

*stop meuservico*

*start meuservico*
*status meuservico*

Não é preciso fazer o tratamento dos parâmetros passados.
Entre as dezenas de eventos gerados pelo sistema, os principais são:

- control-alt-delete
- filesystem
- net-device-up IFACE=lo
- local-filesystems
- mounting TYPE=nfs4
- mounted MOUNTPOINT=/dev
- mounted MOUNTPOINT=/tmp
- mounted MOUNTPOINT=/var/run TYPE=tmpfs
- runlevel [06]
- runlevel [2345]
- started portmap
- mounting TYPE=nfs
- starting gdm
- starting mountall
- starting network-interface
- starting rc RUNLEVEL=[06]
- starting udev
- startup
- stopped gdm EXIT_STATUS=[!0]
- stopped mountall EXIT_STATUS=[!4]
- stopped mountall EXIT_STATUS=4
- stopped rc RUNLEVEL=[2345]
- virtual-filesystems

Um evento pode ser gerado com o comando initctl:

*initctl emit evento1*

É possível ainda ativar o monitoramento do serviço, fazendo com que o serviço seja reiniciado em caso de parada abrupta. Pra isso, basta colocar no arquivo *.conf* o parâmetro *respawn.*
Segue um exemplo completo de um arquivo de configuração para o Upstart:

```
start on filesystem and net-device-up IFACE=lo
stop on runlevel [016]

respawn

pre-start script
 mkdir /tmp/meuservico
end script

post-stop script
 rm -rf /tmp/meuservico
end script

exec /usr/bin/meuservico.sh
```