Aula 12

12 - O sistema de inicialização

A aplicação de desligamento e carregamento do sistema tem passos e tarefas bem definidas. Afinal, este processo deve ser bem controlado para que o sistema esteja apto tanto para funcionar quanto para ser desativado sem traumas. Um exemplo disso pode ser visto quando se desliga de maneira correta todo o sistema, não comprometendo nenhuma informação e aplicação.

Em linhas gerais, quando ligamos o computador, seguimos por uma série de execuções que vão desde a habilitação do hardware até o funcionamento do software do sistema operacional.

Para qualquer tipo de sistema, em uma plataforma Intel tradicional, ao se iniciar a alimentação do computador, o BIOS (Basic Input/Output System) é quem assume o controle. A partir daí esse BIOS tem uma sequência de instruções que vão desde o reconhecimento e validação do hardware do sistema até a chamada de um carregador de sistema operacional.

O BIOS, após validar o hardware da máquina, procura em sua lista de dispositivos, por algum carregador de sistema operacional nas trilhas iniciais (o que popularmente chamamos de trilha zero) destes dispositivos. Ao encontrar algum carregador, o BIOS passa a execução do hardware para este carregador. Este por sua vez tem a função trazer à memória alguma imagem de sistema operacional que passará a ser o kernel ativo. Os carregadores mais utilizados em Linux são o Grub e o Lilo.

Quando este kernel entra em processamento, podemos dizer que a partir daí o sistema operacional está funcionando e o controle da máquina e dos recursos passam então para sua responsabilidade.

Até esse momento, para todo e qualquer sistema rodando em arquiteturas PC Intel, temos esta lista de passos seguida. A partir daí, é que podemos distinguir a diferença entre sistemas, pois cada um assume um processo de inicialização próprio.

Em se tratando de Unix ou Linux, existem dois tipos de inicialização distintos. Esses estilos de inicializações acompanham as vertentes existentes dos antecessores Unix: Sys V ou BSD. Os dois tipos de inicialização são sucintamente descritos a seguir.

12.1 - SYSV init

O estilo de inicialização do Sys V é o mais utilizado em todos os Linux e não é por acaso. Ele possui inúmeros recursos que o tornam muito mais flexível que o init do BSD.

Após o computador ter sido ligado e o BIOS ter finalizado sua tarefa, entra em cena o carregador de sistema operacional, que na maioria dos casos tem sido a aplicação GRUB. Este programa tem a função de escolher uma imagem de sistema operacional (kernel) compactada no disco para carregá-la na memória principal.

Depois do carregamento na memória, essa imagem torna-se ativa e a partir daí tem o controle total das execuções. Os passos da inicialização no *System V* podem ser resumidos em:

1. O primeiro processo é levantado. Ele tem o nome de *init*. Este processo lê o arquivo */etc/inittab* e define o nível de execução do sistema. Pode-se verificar que este é realmente o processo inicial do sistema através do comando *pstree*.

Existem vários níveis de execução do sistema. Cada distribuição pode eleger os níveis de forma personalizada, mas para sistemas Red Hat/Fedora, eles vão de 0 a até 6. Sendo eles:

-0 Desligar
-1 Monousuário
-2 Multiusuário sem NFS
-3 Multiusuário com NFS
-4 Reservado
-5 Modo gráfico
-6 Reinicio

2. Após determinar o nível de execução o processo init executa o script */etc/rc.d/rc.sysinit.*

3. Se existir o script *rc.serial*, então o mesmo é executado pelo script *rc.sysinit.*

4. O proceso *init* volta ao foco principal e executa o processo */etc/rc.d/rc* com o parâmetro de entrada do valor referente ao seu nível de execução. Isto é, se o nível de execução for 3, então o processo */etc/rc.d/rc 3* é chamado para configurar o sistema.

5. No comando das tarefas, o script *rc,* por sua vez, aciona todos outros scripts de um dos diretórios de nível de execução referenciado. Ex:*/etc/rc5.d/*, */etc/rc3.d* , entre outros. É importante ressaltar que todos esses scritps dos diretórios *rc[nível].d* são links simbólicos para os scripts existentes no diretório */etc/init.d/*.

Tomando como exemplo o arquivo que inicializa o sistema *gdm*, no nível de execução 3, certamente haveria dentro do diretório /etc/rc3.d/ um link simbólico, tal como:

S30gdm -> ../init.d/gdm

Além disso, cada um dos scripts existentes nesses diretórios de *rc[nível].d* tem uma denominação especial. Se começados pela letra *S* significam que serão acionados com parâmetro *start* para iniciar um serviço. Se começados pela letra *K* significam que serão acionados para desligar um serviço. Portanto os scripts em */etc/init.d/* tem a possibilidade tanto de iniciar ou de para um serviço, basta acioná-los como o parâmetro *start* ou *stop*. Ex: */etc/init.d/network stop*

6. Após a execução de todos scripts de serviços, o *init* volta ao papel principal e invoca o script inicial de serviços locais, contidos no arquivo */etc/rc.loca*l

7. Depois de ter invocado todos esses serviços, o *init* invoca, por fim, os processos de *mingetty* para ativar os terminais.

Um exemplo de arquivo */etc/inittab* pode ser visto a seguir:

```
id:2:initdefault:
# Boot-time system configuration/initialization script.
si::sysinit:/etc/rc.sysinit
# What to do in single-user mode.
~:S:wait:/sbin/sulogin
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
l0:0:wait:/etc/rc 0
l1:1:wait:/etc/rc 1
l2:2:wait:/etc/rc 2
l3:3:wait:/etc/rc 3
l4:4:wait:/etc/rc 4
l5:5:wait:/etc/rc 5
l6:6:wait:/etc/rc 6
```

 # What to do at the "3 finger salute".
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

 # Runlevel 2,3: getty on virtual consoles
 # Runlevel 3: mgetty on terminal (ttyS0) and modem (ttyS1)
1:23:respawn:/sbin/mingetty tty1
2:23:respawn:/sbin/mingetty tty2
3:23:respawn:/sbin/mingetty tty3
4:23:respawn:/sbin/mingetty tty4
S0:3:respawn:/sbin/agetty ttyS0 9600 vt100-nav
S1:3:respawn:/sbin/mgetty -x0 -D ttyS1

Como se pode notar, há uma sintaxe bem definida para as linhas do inittab. Onde cada uma delas indica que o processo *init* deverá realizar segundo um nível de execução estabelecido.

A formação básica de uma entrada do inittab é:

id:*runlevels*:*action*:*process*

id - identificador para uma entrada do *init*. É uma sequência de 4 caracteres quaisquer identifica exclusivamente aquela entrada no *inittab*.

runlevels - níveis de execução que devem ser considerados para aquela ação/processo.

action - ação executada para o processo em questão. A lista de ações permitidas é composta por: *respawn , wait, once, boot, bootwait, off, ondemand, initdefault, sysinit, powerwait, powerfail, powerokwait, powerfailnow, ctrlaltdel e kbrequest .*

process - o programa ou processo em si a ser executado.

Algumas das ações podem determinar que o campo de processo ou de *runlevel* sejam ignorados. Ações primordiais, como por exemplo *initdefault*, que determina qual nível de execução a ser considerado pelo *init*, é por exemplo, uma ação que não exige processo vinculado (vide exemplo de arquivo *inittab*).

12.1.2 - O upstart

O *upstart*, como é chamado, é um *daemon* para inicialização que vem substituindo o tão popular daemon *init*, anteriormente descrito. O *Ubuntu Edgy* e o Fedora 9 passaram a incorporar o *upstart*, assegurando à todas distribuições subsequentes a instalação padrão do sistema de inicialização com *upstart* também.

A diferença crucial entre o antigo *daemon init* e o *upstart* consiste no fato de o *daemon* do *upstart* ser orientado a eventos. Isto é, ele não apenas executa comandos, ele pode também administrar, emitir e processar os sinais (eventos) enquanto ainda executa outras aplicações .

Basicamente, configurar e entender o *upstart*, para quem esta familiarizado ao *System* V é fácil, pois o *upstart* simula a execução de níveis e scripts conforme um *Sysyem* V tradicional.

O que ocorre, no upstart, é que o processo *init* lê uma tabela de eventos conhecidos e programados que estão em algum subdiretório dentro de /etc (em geral /etc/init, /etc/event.d, /etc/apm/event.d). Com isso o *upstart* conhece todos eventos pretendidos e pode tratá-los quando seus sinais forem recebidos.

Pode-se definir o nível padrão de inicialização através do arquivo *rc-default* e através dos arquivos como *rc0, rc1, tty0* pode-se determinar quais ações a serem tomadas ou quais eventos podem ser executados para cada um dos níveis ou terminais.

Exemplo de um diretório */etc/apm/event.d* pode ser vista abaixo:

```
-rw-r--r--  1 root root  260 2008-09-29 20:52 control-alt-delete
-rw-r--r--  1 root root  187 2009-04-08 22:16 last-good-boot
-rw-r--r--  1 root root  299 2008-09-29 20:52 logd
-rw-r--r--  1 root root  552 2008-09-29 20:52 rc0
-rw-r--r--  1 root root  342 2008-09-29 20:52 rc1
-rw-r--r--  1 root root  403 2008-09-29 20:52 rc2
-rw-r--r--  1 root root  403 2008-09-29 20:52 rc3
-rw-r--r--  1 root root  403 2008-09-29 20:52 rc4
-rw-r--r--  1 root root  403 2008-09-29 20:52 rc5
-rw-r--r--  1 root root  422 2008-09-29 20:52 rc6
-rw-r--r--  1 root root  485 2008-09-29 20:52 rc-default
-rw-r--r--  1 root root  392 2008-09-29 20:52 rcS
-rw-r--r--  1 root root  575 2008-09-29 20:52 rcS-sulogin
-rw-r--r--  1 root root  558 2008-09-29 20:52 sulogin
-rw-r--r--  1 root root  306 2008-09-29 20:52 tty1
-rw-r--r--  1 root root  300 2008-09-29 20:52 tty2
-rw-r--r--  1 root root  300 2008-09-29 20:52 tty3
-rw-r--r--  1 root root  300 2008-09-29 20:52 tty4
-rw-r--r--  1 root root  300 2008-09-29 20:52 tty5
-rw-r--r--  1 root root  300 2008-09-29 20:52 tty6
```

Onde o arquivo de maior interesse é aquele que invoca o *rc* padrão. No caso de termos o nível de execução (*rc*) padrão como número 2, seria importante analisar o conteúdo de *rc2*. Nele podemos encontrar:

```
# rc2 - runlevel 2 compatibility
#
# This task runs the old sysv-rc runlevel 2 ("multi-user") scripts.  It
# is usually started by the telinit compatibility wrapper.

start on runlevel 2
stop on runlevel [!2]
console output
script
    set $(runlevel --set 2 || true)
    if [ "$1" != "unknown" ]; then
       PREVLEVEL=$1
       RUNLEVEL=$2
       export PREVLEVEL RUNLEVEL
    fi
    exec /etc/init.d/rc 2
end script
```

O arquivo *rc2*, de fato, invoca o script */etc/rc* em seu nível de execução 2, fazendo os passos já tradicionais de inicialização do sys V. Isto é feito para manter a familiaridade e compatibilidade com os sistemas de inicialização já existentes.

O *upstart* ainda não se tornou um padrão em sistema de inicialização, mas tem sido visto com bons olhos por várias distribuições. Dentro em breve, é provável que apenas um entre o *System V* init e o *Upstart* mantenha-se como referência em sistemas de inicialização.

12.2 - BSD init

O processo do BSD é muito mais simplificado. Ao invés do arquivo *inittab* existe o arquivo */etc/rc* que é invocado pelo processo *init*.

A inicialização neste método não possui níveis de execução, mas sim, somente scripts de inicialização e parada de serviços. Todos estes scripts ficam localizados em /*etc* ou */etc/rc.d.*

Após a execução destes scripts ainda são acionados outros dois, assim como no *System V*, existem o */etc/rc.serial* e o */etc/rc.local*.

Atualmente, é difícil encontrarmos o BSD *init* puro nos sistemas Linux. A evolução do init Sys V acabou trazendo benefício que o elegem em detrimento ao BSD clássico, que é bem simplificado. Alguns sistemas ainda mantém inits BSD, mas estes também incorporam algumas das funcionalidades e compatibilidades de sys V.

12.3 - Bibliografia

Este documento é resenha retirada do capítulo 7 de: Ferreira, E. Rubem. Guia do Administrador Linux. 1 edição, 2003, Novatec Editora, São Paulo.