

Startup Scripts

- After single-user mode prompt, **init** runs startup scripts.
- Shell scripts to configure and initialize system services.
- Sets name of computer.
- sets timezone.
- Check disks with **fsck**.

- Mounts systems disks.
- Remove old files from **/tmp**.
- Configure network interfaces.
- Start up daemons and network services.
- Separate configuration files for scripts.

Run Levels

- **init** defines 7 run levels.

- 10 possible.

Level 0 Shut down.

Level 1 or S Single-user.

Level 2-5 Multiuser levels.

Level 6 Reboot.

- Run Level 5 for X Windows.

- Run Level 4 rarely used.
- Can't remain in Levels 0 or 6.
- Normal 2 or 3.
- More than necessary.
- Phone switch has 7 run levels?

/etc/inittab

- Tells **init** what do to at each run level.
- Defines commands that are run when system enters particular run level.
- Default run level set there.
- Progresses upward through run levels on boot.
- Progresses down through run levels on shutdown.

rc directories

- Change run level scripts.
- **/etc/init.d/rc**
- Contains scripts that are called from **inittab**.
- These scripts execute other scripts from run-level-specific directories (**/etc/rc.d**).
- Master copies of start-up scripts in **/etc/init.d**.

- Each script responsible for one daemon or service.
- Arguments of **start**, **stop**, or **restart**.
- Manually start and stop by running scripts.

Example

```
#!/bin/bash
#
# Init file for OpenSSH server daemon
#
# chkconfig: 2345 55 25
# description: OpenSSH server daemon
#
# processname: sshd
# config: /etc/ssh/ssh_host_key
# config: /etc/ssh/ssh_host_key.pub
# config: /etc/ssh/ssh_random_seed
# config: /etc/ssh/sshd_config
```

```
# pidfile: /var/run/sshd.pid

# source function library
. /etc/rc.d/init.d/functions

# pull in sysconfig settings
[ -f /etc/sysconfig/sshd ] && . /etc/sysconfig/sshd

RETVAL=0
prog="sshd"

# Some functions to make the below more readable
KEYGEN=/usr/bin/ssh-keygen
SSHD=/usr/sbin/sshd
RSA1_KEY=/etc/ssh/ssh_host_key
```

```
RSA_KEY=/etc/ssh/ssh_host_rsa_key
DSA_KEY=/etc/ssh/ssh_host_dsa_key
PID_FILE=/var/run/sshd.pid
```

```
do_rsa1_keygen() {
if [ ! -s $RSA1_KEY ]; then
echo -n $"Generating SSH1 RSA host key: "
if $KEYGEN -q -t rsa1 -f $RSA1_KEY -C '' -N '' >&/dev/null; then
chmod 600 $RSA1_KEY
chmod 644 $RSA1_KEY.pub
success $"RSA1 key generation"
echo
else
failure $"RSA1 key generation"
echo
```

```
exit 1
```

```
fi
```

```
fi
```

```
}
```

```
do_rsa_keygen() {
```

```
if [ ! -s $RSA_KEY ]; then
```

```
echo -n "Generating SSH2 RSA host key: "
```

```
if $KEYGEN -q -t rsa -f $RSA_KEY -C '' -N '' >&/dev/null; then
```

```
chmod 600 $RSA_KEY
```

```
chmod 644 $RSA_KEY.pub
```

```
success "RSA key generation"
```

```
echo
```

```
else
```

```
failure "RSA key generation"
```

```
echo
exit 1
fi
fi
}
```

```
do_dsa_keygen() {
if [ ! -s $DSA_KEY ]; then
echo -n "Generating SSH2 DSA host key: "
if $KEYGEN -q -t dsa -f $DSA_KEY -C '' -N '' >&/dev/null; then
chmod 600 $DSA_KEY
chmod 644 $DSA_KEY.pub
success "DSA key generation"
echo
else
```

```
failure $"DSA key generation"  
echo  
exit 1  
fi  
fi  
}
```

```
do_restart_sanity_check()  
{  
$SSHHD -t  
RETVAL=$?  
if [ ! "$RETVAL" = 0 ]; then  
failure $"Configuration file or keys are invalid"  
echo  
fi  
}
```

```
}
```

```
start()
```

```
{
```

```
# Create keys if necessary
```

```
do_rsa1_keygen
```

```
do_rsa_keygen
```

```
do_dsa_keygen
```

```
echo -n "$Starting $prog:"
```

```
initlog -c "$SSHD $OPTIONS" && success || failure
```

```
RETVAL=$?
```

```
[ "$RETVAL" = 0 ] && touch /var/lock/subsys/sshd
```

```
echo
```

```
}
```

```
stop()
{
echo -n $"Stopping $prog:"
killproc $SSHD -TERM
RETVAL=$?
[ "$RETVAL" = 0 ] && rm -f /var/lock/subsys/sshd
echo
}
```

```
reload()
{
echo -n $"Reloading $prog:"
killproc $SSHD -HUP
RETVAL=$?
```

```
echo
```

```
}
```

```
case "$1" in
```

```
start)
```

```
start
```

```
;;
```

```
stop)
```

```
stop
```

```
;;
```

```
restart)
```

```
stop
```

```
start
```

```
;;
```

```
reload)
```

```
reload
;;
condrestart)
if [ -f /var/lock/subsys/sshd ] ; then
do_restart_sanity_check
if [ "$RETVAL" = 0 ] ; then
stop
# avoid race
sleep 3
start
fi
fi
;;
status)
status $SSHHD
```

```
RETVAL=$?
```

```
;;
```

```
*)
```

```
echo $"Usage: $0 {start|stop|restart|reload|condrestart|status}"
```

```
RETVAL=1
```

```
esac
```

```
exit $RETVAL
```

rc directories

- **init** needs additional information about which scripts to run.
- Looks in **rc/level.d**
- They contain symbolic links to scripts in **init.d**.
- Symbolic links start with **S** or **R** followed by number and name of service.
- **init** runs all the appropriate scripts when it makes a transition for one run-level to another.

- Place symbolic link in appropriate directory to add service or daemon.
- **In -s /etc/init.d/sshd /etc/rc2.d/S99sshd**

Redhat

- **rc.local** is last script run.
- System specific information.
- Overwritten at install.
- **chkconfig**: command line tool for maintaining symbolic links in **rc#.d** directories.
- Configuration files in **/etc/sysconfig**.

File/Dir	Functions or Contents
apmd	Lists arguments for Advanced Power Management
clock	Type of clock
console	always empty
hwconf	Hardware info. used by kudzu.
i18n	System local settings.
init	Way messages from startup scripts displayed.
keyboard	Sets keyboard type.
mouse	Sets mouse type.
network	Sets global network options.
network-scripts	Accessory scripts and config files.

`/etc/inittab`

- Lines as **id:runlevels:action:process**
- Empty and lines beginning with `#` are ignored.
- `id`: Identifies line in the file. Should be unique. For **getty** specifies terminal.
- `runlevels`: Run levels line is considered for.
- `action`: Action to take. **respawn** to run command in next field when it exits. **once** to run once.

- process: command to run.

- **wait** run command once and wait for completion.

Rebooting and Shutting Down

- Reboot as last resort.
- Linux filesystems buffer changes in memory and write to disk only sporadically.
- Faster but vulnerable to losing data.
- UNIX sensitive to shutting down properly.

shutdown

- safest, most considerate, and thorough
- waits before shutting down, send message by **wall** to users.
- **-h** halt or **-r** reboot
- **-F** fsck or **f** not (default)
- **shutdown -h 09:30 "Going down for scheduled maintenance"**

- `shutdown -h +15` "Going down for scheduled maintenance"

halt, reboot, and telinit

- **shutdown -h**: essential things to bring system down: logs, kills processes, executes **sync**, waits for filesystem writes to complete, and halts kernel.
- **halt -n** prevents **sync**
- **shutdown -r**: essential things to bring system down: logs, kills processes, executes **sync**, waits for filesystem writes to complete, and halts kernel, reboots. Also, supports **-n** flag.
- Change run-level. No warning message or grace period.

Rootly Powers

- System files and processes are owned by fictitious user called “root.”
- Owner can modify.
- UID 0
- Other users also, but bad idea.
- root can change its UID and GID (example login)

root

- Change the root directory
- create device files
- set system clock
- raise resource limits and process priorities
- set systems hostname

- configure network interfaces
- open privileged ports (< 1024)
- shutdown system.
- Choose good root password.
- **SU** to root.

Other Pseudo-Users

bin Legacy owner of system commands.

daemon Owner of unprivileged system software.

nobody Generic NFS user.