Solaris init states refer to the level of services provided by the system. The exact services and processes run at each init level are determined by the scripts in the /etc/rc#.d directories. The default service levels for each init state are listed below:

- **0**: The system is at the PROM monitor (`ok>`) or security monitor (`>`) prompt. It is safe to shut down the system when it is at this init state.
- **1, s or S**: This state is known as "single-user" or "system administrator" mode. Root is the only user on the system, and only basic kernel functions are enabled. A limited number of filesystems (usually only root and `/usr`) are mounted. This init state is often used for sensitive functions (such as kernel libc patches) or while troubleshooting a problem that is keeping the system from booting into multiuser mode.
- **2**: Multiple users can log in. Most system services (except for NFS server and printer resource sharing) are enabled.
- **3**: Normal operating state. NFS and printer sharing is enabled, where appropriate.
- **4**: Usually undefined.
- **5**: Associated with the `boot -a` command. The system is taken to init 0 and an interactive boot is started.
- **6**: Reboot. This state takes the system to init state 0 and then to the default init state (usually 3, but can be redefined in the `/etc/inittab` file).

The init states are defined in the `/etc/inittab` file, which usually points at the scripts in `/sbin/rc`*`run-level`*. These scripts in turn examine the contents of the `/etc/rc`*`run-level`* directories. The scripts in these directories whose names begin with the letter `K` are run in "stop" mode first in alphabetical order. Then the scripts whose names begin with the letter `S` are run in "start" mode in alphabetical order.

To get to a desired run level n, each of the rc (run control) scripts from 1 to n is run. To get to run level 0, the `K` scripts are run in each `rc#.d` directory between the current run level and 0 in reverse numerical order.

In the default configuration, the rc scripts accomplish the following tasks:

- `/sbin/rc0`
  - Stop system services/daemons.
  - Terminate running processes.
  - Unmount all file systems.
- `/sbin/rc1`
  - Stop system services/daemons.
  - Terminate running processes.
  - Unmount all file systems.
  - Bring up the system in single-user mode.
- `/sbin/rc2`
  - Set the `TIMEZONE` variable.

    - Stop the print and NFS services.
    - Stop the `vold` daemon.
    - Mount local filesystems, enable disk quotas (as appropriate).
    - Remove temporary files.
    - Create new device entries if this is the result of a `boot -r`.
    - Save a core file if enabled.
    - Configure system accounting, (as appropriate).
    - Set the default router.
    - Set the NIS domain.
    - Set up the network interfaces appropriately.
    - Start `inetd`.
    - Start `named`, if appopriate.
    - Start `rpcbind`.
    - Start `kerbd` (the Kerberos client daemon) if appropriate.
    - Start `ypbind` or `rpc.nisd` as appropriate.
    - Start `keyserv`.
    - Start `statd` and `lockd`.
    - Mount NFS filesystems from `/etc/vfstab`.
    - Start the automounter.
    - Start `cron`.
    - Start `lp` daemons, as appropriate.
    - Start `sendmail`.
- `/sbin/rc3`
    - Clean up `sharetab`.
    - Start `nfsd` and `mountd`.
    - Start `rarpd` and `rpc.bootparamd`, as appropriate.
- `/sbin/rc4` is usually not defined. It can be used in a non-default configuration to achieve a tailored run level.
- `/sbin/rc5`
    - Kill print daemons.
    - Unmount local file systems.
    - Kill `syslogd`.
    - Unmount NFS file systems.
    - Stop NFS services.
    - Stop NIS services.
    - Stop RPC services.
    - Stop `cron` services.
    - Stop `statd` and `lockd` (NFS client services).
    - Kill active processes.
    - Initiate an interactive boot.
- `/sbin/rc6`
    - Stop system services/daemons.
    - Terminate running processes.

    - Unmount all file systems.
    - Boots to the `initdefault` level from the `/etc/inittab`
- `/sbin/rcS`: This run level differs from 1 in the following particulars:
    - Minimal network is established.
    - System name is set.
    - root, `/usr` and `/usr/kvm` filesystems are checked and mounted (if necessary).
    - Pseudo file systems `proc` and `/dev/` are started.
    - Rebuilds device entries (for reconfiguration reboots only).

Solaris Run level change

I need to find out runlevel related information, as you may need to change runlevel for following causes:

1. Halt/reboot system when shutdown command don't work
2. Troubleshooting or repairing system

The first thing I noticed was /etc/inittab file is different from the Linux version. However, runlevel are quite identical

Default Solaris Run Level

- S : Single user state (useful for recovery)
- 0 : Access Sun Firmware (ok> prompt)
- 1 : System administrator mode
- 2 : Multi-user w/o NFS
- 3 : Multi-user with NFS
- 4 : Unused
- 5 : Completely shutdown the host (like performing a power-off @ OBP) [thanks to Marco]
- 6 : Reboot but depend upon initdefault entry in /etc/inittab

Solaris Find out runlevel

To find out current runlevel use who command:

$ who -r

Output:

```
  .       run-level 3  Mar  3 14:04     3      0  S
```

Solaris changing runlevels after bootup

You need to use init command, for example change runlevel to 2.

/sbin/init 2

Solaris changing the default runlevel

An entry with initdefault (in /etc/inittab file) is scanned only when init is initially invoked. init uses this entry to determine which run level to enter initially.

Open /etc/inittab file:

vi /etc/inittab

Find out this entry:

is:3:initdefault:

Change ***is:3*** to number you want, don't use S, 0, 6 ;). Save file.

Recommended Links

- Read man page of inittab and init for more information.
- There is also a nice program called runlevel for Solaris.

NAME

init, telinit - process control initialization

SYNOPSIS

/sbin/init [0123456abcQqSs]

/etc/telinit [0123456abcQqSs]

AVAILABILITY

SUNWcsr

DESCRIPTION

init is a general process spawner. Its primary role is to create processes from information stored in the file /etc/inittab.

At any given time, the system is in one of eight possible run levels. A run level is a software configuration under which only a selected group of processes exists. Processes spawned by init for each of these run levels are defined in /etc/inittab. init can be in one of eight run levels, 0 - 6 and S or s (S and s are identical). The run level changes when a privileged user runs /sbin/init. This sends appropriate signals to the original init spawned by the operating system at boot time, saying which run level to invoke.

When the system is booted, init is invoked and the following occurs. First, it reads /etc/default/init to set environment variables. This is typically where TZ (time zone) and locale-related environments such as LANG or LC_CTYPE get set.

init then looks in /etc/inittab for the initdefault entry (see inittab(4)). If one exists, init usually uses the run level specified in that entry as the initial run level to enter. If there is no initdefault entry in /etc/inittab, init asks the user to enter a run level from the virtual system console. If an S or s is entered, init goes to the single-user state. In this state, the virtual console terminal is assigned to the user's terminal and is opened for reading and writing. The command /sbin/su is invoked and a message is generated on the physical console saying where the virtual console has been relocated. Use either init or telinit to change the run level of the system. Note that if the shell is terminated (using an end-of-file), init only re-initializes to the single-user state if /etc/inittab does not exist.

If a 0 through 6 is entered, init enters the corresponding run level. Run levels 0, 5, and 6 are reserved states for shutting the system down. Run levels 2, 3, and 4 are available as multi-user operating states.

If this is the first time since power up that init has entered a run level other than single-user state, init first scans /etc/inittab for boot and bootwait entries (see inittab(4)). These entries are performed before any other processing of /etc/inittab takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. init then scans /etc/inittab and

executes all other entries that are to be processed for that run level.

To spawn each process in /etc/inittab, init reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by /etc/inittab, init waits for one of its descendant processes to die, a powerfail signal, or a signal from another init or telinit process to change the system's run level. When one of these conditions occurs, init re-examines /etc/inittab.

New entries can be added to /etc/inittab at any time; however, init still waits for one of the above three conditions to occur before re-examining /etc/inittab. To get around this, init Q or init q command wakes init to re-examine /etc/inittab immediately.

When init comes up at boot time and whenever the system changes from the single-user state to another run state, init sets the ioctl(2) states of the virtual console to those modes saved in the file /etc/ioctl.syscon. This file is written by init whenever the single-user state is entered.

When a run level change request is made, init sends the warning signal (SIGTERM) to all processes that are undefined in the target run level. init waits five seconds before forcibly terminating these processes by sending a kill signal (SIGKILL).

When init receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in /var/adm/utmp and /var/adm/wtmp if it exists (see who(1)). A history of the processes spawned is kept in /var/adm/wtmp.

If init receives a powerfail signal (SIGPWR) it scans /etc/inittab for special entries of the type powerfail and powerwait. These entries are invoked (if the run levels permit) before any further processing takes place. In this way init can perform various cleanup and recording functions during the powerdown of the operating system.

telinit

telinit, which is linked to /sbin/init, is used to direct the actions of init. It takes a one-character argument and signals init to take the appropriate action.

OPTIONS

0 Go into firmware.

1 Put the system in system administrator mode. All file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

2 Put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.

3 Start the remote file sharing processes and daemons. Mount and advertise remote resources. Run level 3 extends multi-user mode and is known as the remote-file-sharing state.

4 Is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.

5 Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible.

6 Stop the operating system and reboot to the state defined by the initdefault entry in /etc/inittab.

a, b, c process only those /etc/inittab entries having the a, b, or c run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

Q, q Re-examine /etc/inittab.

S, s Enter single-user mode. When this occurs, the terminal which executed this command becomes the system console. This is the only run level that doesn't require the existence of a properly formatted /etc/inittab file. If this file does not exist, then by default, the only legal run level that init can enter is the single-user mode. When the system comes up to S or s, file systems for users' files are not mounted and only essential kernel processes are running. When the system comes down to S or s, all mounted file systems remain mounted, and all processes started by init that should only be running in multi-user mode are killed. In addition, any process that has a utmp entry will be killed. This last condition insures that all port monitors started by the SAC are killed and all services started by these port monitors, including ttymon login services, are killed. Other processes not started directly by init will remain running. For example, cron remains running.

FILES

/etc/inittab	controls process dispatching by init
/var/adm/utmp	accounting information
/var/adm/wtmp	history of all logins since file was last created
/etc/ioctl.syscon	
/dev/console	system console device
/etc/default/init	environment variables. Default values can be set for the following flags in /etc/default/init. For example: TZ=US/Pacific
	TZ Either specifies the timezone information (see ctime(3C)) or the name of a timezone information file

	/usr/share/lib/zoneinfo.
LC_CTYPE	Character character-ization information.
LC_MESSAGES	Message translation.
LC_MONETARY	Monetary formatting information.
LC_NUMERIC	Numeric formatting information.
LC_TIME	Time formatting information.
LC_ALL	If set, all other LC_* environmental variables take-on this value.
LANG	If LC_ALL is not set, and any particular LC_* is also not set, the value of LANG is used for that particular environmental variable.

SEE ALSO

login(1), sh(1), stty(1), who(1), shutdown(1M), ttymon(1M), kill(2), ctime(3C), inittab(4), utmp(4), utmpx(4), termio(7)

DIAGNOSTICS

If init finds that it is respawning an entry from /etc/inittab more than ten times in two minutes, assumes that there is an error in the command string in the entry, and generates an error message on the system console. It will then refuse to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned init or telinit. This prevents init from eating up system resources when someone makes a typographical error in the inittab file, or a program is removed that is referenced in /etc/inittab.

When attempting to boot the system, failure of init to prompt for a new run level may be caused by the virtual system console being linked to a device other than the physical system console.

NOTES

init and telinit can be run only by a privileged user.

The S or s state must not be used indiscriminately in /etc/inittab. When modifying this file, it is best to avoid adding this state to any line other than initdefault.

If a default state is not specified in the initdefault entry in /etc/inittab, state 6 is entered. Consequently, the system will loop by going to firmware and rebooting continuously.

If the utmp file cannot be created when booting the system, the system will boot to state s regardless of the state specified in the initdefault entry in /etc/inittab. This can occur if the /var file system is not accessible.