

# Back to the Basics: Solaris Default Processes and init.d Pt. I

Hal Flynn 2000-05-29

## Introduction

This article has been written to provide insight into a stock installation of Solaris 8, and the services started by default. Solaris 8 by default runs many services. This example was provided using Solaris 8, which is the latest version available, and a Sparcstation 20. Most of this document will apply to releases of Solaris prior to 8, and to both the Sparc and Intel architectures. For documentation purposes, a full OEM install was done. Many topics discussed will be familiar to seasoned administrators. However, this document will benefit all parties involved in the administration and security aspects of Solaris.

## The Install

If you're familiar with the installation process of Solaris, you might want to skip to the next section.

The initial lab system used for this document is a Sparcstation 20 with 224 Megs of ram, 2 SCSI Disks (5 Gig and 1 Gig), and dual SM61 processors. A tad sluggish under 8, but sufficient. The system was additionally installed on an isolated network (for security purposes, of course).

Thebes (the name of the system) was configured using a 24 bit subnet, and the ip address 192.168.1.2. IP version 6 compatibility was additionally enabled. It was not configured using dhcp under Solaris (something new with 8). However, something interesting occurred during the install using dhcp.

Solaris 8 now has a new selection in the window feature, which allows the person installing the system to select dhcp to give the system its configuration information. The isolated network was using dhcpd 3.0 b1p113 to bring up most of the workstations, and giving the machines on the network router, nameservers, and ip address (codes 1, 3, 6, and 15 per RFC 2132). After dhcpd received the request from thebes, and the mac entry was located and data returned, thebes crashed. This was reproduced three out of three times. To work around the problem, thebes was given the information statically.

An initial install of the OS was done. Kerberos (part of the Sun Enterprise Authentication Mechanism (SEAM) package) was not configured. No additional geographic region support was selected. The OS was installed on the 5 Gig disk (only ~1 Gig is required for a full install of 8), and /export/home was installed on the 1 Gig disk.

## Our example

Using the `ps` command with the `e` and `f` flags, I was able to get a list of all running processes. I've built

our example off this. *ps* gives the current reports the process status of the system. The *e* flag in *ps* prints everything running on the system at time of execution, the *f* flag prints full status information.

## • The Output

After running the command on our freshly installed, unaltered Solaris 8 system, this is the full output of *ps -ef*, which represents everything running on the system.

```
$ ps -ef |more
  UID    PID  PPID  C   STIME TTY      TIME  CMD
  root     0     0  0 02:20:10 ?        0:01  sched
  root     1     0  0 02:20:13 ?        0:01  /etc/init -
  root     2     0  0 02:20:13 ?        0:00  pageout
  root     3     0  0 02:20:13 ?        7:39  fsflush
  root    282     1  0 02:21:19 ?        0:00  /usr/lib/saf/sac -t 300
  root    143     1  0 02:21:05 ?        0:00  /usr/sbin/rpcbind
  root     52     1  0 02:20:24 ?        0:00  /usr/lib/devfsadm/devfseventd
  root     56     1  0 02:20:39 ?        0:00  /usr/lib/devfsadm/devfsadmd
  root    122     1  0 02:21:04 ?        0:00  /usr/sbin/in.routed -q
  root    178     1  0 02:21:07 ?        0:00  /usr/lib/nfs/lockd
  root    129     1  0 02:21:05 ?        0:00  /usr/lib/inet/in.ndpd
  root    212     1  0 02:21:10 ?        0:00  /usr/sbin/nscd
  root    218     1  0 02:21:11 ?        0:00  /usr/lib/lpsched
  root    166     1  0 02:21:07 ?        0:01  /usr/sbin/inetd -s
  root    197     1  0 02:21:09 ?        0:00  /usr/sbin/syslogd
  root    186     1  0 02:21:08 ?        0:01  /usr/lib/autofs/automountd
daemon  174     1  0 02:21:07 ?        0:00  /usr/lib/nfs/statd
  root    248     1  0 02:21:13 ?        0:00  /usr/sbin/vold
  root    201     1  0 02:21:09 ?        0:00  /usr/sbin/cron
  root    231     1  0 02:21:12 ?        0:00  /usr/lib/power/powerd
  root    243     1  0 02:21:12 ?        0:00  /usr/sadm/lib/wbem/cimombot start
  root    241     1  0 02:21:12 ?        0:00  /usr/lib/utmpd
  root   1780    276  0 01:37:50 ?        0:03  /usr/openwin/bin/Xsun :0 -nobanne
r -auth /var/dt/A:0-K0aGIa
  root   1795   1781  0 01:37:53 ?        0:01  dtgreet -display :0
  root    268     1  0 02:21:15 ?        0:00  /usr/lib/snmp/snmpdx -y -c /etc/s
nmp/conf
  root    283     1  0 02:21:19 console  0:00  /usr/lib/saf/ttymon -g -h -p theb
es console login: -T sun -d /dev/console -l c
  root    271    268  0 02:21:16 ?        1:18  mibiisa -r -p 32792
```

```

root    278      1  0 02:21:18 ?          0:00 /usr/lib/dmi/snmpXdmid -s thebes
root   1783      1  0 01:37:52 ?          0:00 /usr/openwin/bin/fbconsole -d :0
root    276      1  0 02:21:18 ?          0:00 /usr/dt/bin/dtlogin -daemon
root    277      1  0 02:21:18 ?          0:00 /usr/lib/dmi/dmispd
root    287     282  0 02:21:20 ?          0:00 /usr/lib/saf/ttymon
root    304      1  0 02:22:13 ?          0:01 /usr/lib/sendmail -bd -q15m
root   1781     276  0 01:37:50 ?          0:00 /usr/dt/bin/dtlogin -daemon
root    372     166  0 02:24:49 ?          0:00 rpc.ttdbserverd
root   1805     166  0 01:39:30 ?          0:00 rquotad

```

## • Analysis of the table

- *sched* is the first process running. It is referred to as the swapper. This process is responsible for operating system scheduling, and swapping out light weight processes when necessary to run higher priority processes. From this process, the scheduling of and swapping of processes on the system is controlled.
- *init* is the process that is responsible for the execution of all processes at their respective run levels. At bootstrap time, *init* is the first process started. From its execution, *init* starts all other processes, and brings the machine to its default run level (for Solaris, this is run level 3).
- *pageout* is the next process in the sequence. It is used to control the paging out of memory to disk, and back in again.
- *fsflush* is a daemon responsible for writing data back to the disks. The kernel checks superblocks on a 30 second interval, and the data in the superblock is either idle or unchanged, the kernel uses *fsflush* to clear the superblock and send the information back to the disks.
- *sac* is the Service Access Controller, and is started when the system enters multiuser mode. *sac* is a program designed to watch ports on a Solaris system. It can provide statics on port use, poll for failure, restart port monitors that fail, and a variety of other functions.
- *fbconsole* is used to redirect `/dev/console` output to a console window in X.
- *rpcbind* is the rpc program port mapper. This program runs on port 111, and contains a directory of all rpc services running on a system. We'll not get too deep into this in this section yet, but sufficed it to say that if you're making use of programs that use rpc, this daemon is mandatory. If you know little or nothing about remote procedure call and what

it's used for, one is probably better off without it and `rpc` altogether.

- `devfseventd` and `devfsadmd` we'll cover together, as they're dependent upon one another. These two daemons are part of the Solaris 8 device management package. `devfseventd` is the kernel event notification daemon. This daemon runs on the system, monitoring the kernel for things such as when device nodes are added and removed from the kernel device tree. `devfsadmd` is the replacement to the antiquated programs such as `disks`, `tapes`, and `devlinks` (all part of the old `devfs` tools). `devfsadmd` builds the links in the `/dev` and `/devices` directories. All the old `devfs` tools are now symbolic links to `devfsadm`.
- `in.routed` is the routing daemon. This daemon listens to UDP on port 520 for dynamic updates to the routing table. This daemon makes use of Routing Information Protocol (RIP). The `q` flag puts this daemon into a passive state, where it listens only for routing updates. Details for RIP can be found in RFC 1723.
- `lockd` is the file locking manager for `nfs`. This daemon controls all the file locking procedures necessary for alteration of files mounted remotely via `nfs`. This daemon is necessary for both client and server running `nfs`. The daemon handles incoming locking requests from other locking daemons for files it's serving. It additionally communicates with other locking daemons to for locking on remotely mounted file systems.
- `in.idpd` is a daemon new in Solaris 8. This daemon provides auto configuration for hosts and routers under IPv6. IPv6 compatibility is new with this release of Solaris.
- `nscd` is the name service cache daemon. This daemon caches the most common name service requests for a system to enhance operation. The daemon caches common things such as password file entries, group file entries, host file entries, and miscellaneous attributes. This is not to be confused with the Domain Name System. The primary function of this service revolves around NIS and NIS+. The operation of this daemon is dictated by `/etc/nscd.conf`
- `lpsched` is a printer service utility. It is part of the UNIX `lp` package, and starts or restarts print services on a printer.
- `inetd` is the services superserver. It provides a wrapper-like service for numerous user services on a Solaris system, starting services only when they're called, and killing them when they're not needed. For more information on `inetd`, see the first two editions of "Back to the Basics," [part I](#) and [part II](#).

- *syslogd* is the system logging daemon. This daemon is responsible for monitoring and logging system events, or sending them to users on the system. *syslogd* is a critical application on every system, and is configurable with the `/etc/syslog.conf` file.
- *automountd* is the autofs mount daemon. This service makes use of RPC to mount and unmount remote file systems.
- *statd* is the status monitoring daemon, and it works in conjunction with *lockd*. This daemon communicates with clients to tell them when the server has rebooted. The daemon operating on clients also communicates with the *statd* process on the server to tell the server when a client has rebooted. The principle purpose for the daemon is tracking client sessions, and recovery post-crash or after a reboot.
- *vold* is the volume manager. This neat little daemon manages the system cdrom and floppy. When media is inserted into either the cdrom or the floppy drive, *vold* goes to work and mounts the media automatically. Configuration information for this utility is in `/etc/vold.conf`.
- *cron* is a system scheduling utility. Cron is capable of executing events for specific users on a predetermined time schedule if entries are made into the users crontab.
- *powerd* is the system power daemon. This daemon performs suspend operations, saving system state and writing it to disk. Upon reboot of the system, the system picks up from state *powerd* saved, and begins a resume operation.
- *cimomboot* is part of the Solaris Web Based Enterprise Management (WBEM). To give a little background, CIM is the Common Information Model, an Object Oriented model built to describe managed resources such as disks and CPUs. *cimomboot* is the daemon process for starting a WBEM session. When a WBEM client requests a connection, *cimomboot* starts the CIM Object Manager. For more in depth information about this service, see the Solaris WBEM Services Administrator Guide at <http://docs.sun.com>. We'll only touch lightly upon it in this document.
- *utmpd* is the *utmp* and *utmpx* monitoring daemon. *utmp* has been obsoleted by *utmpx*, but for reverse compatibility, it exists. To cut to the chase, *utmpx* is used to record the current users on a system. When a user terminates a process or logs out of a system, *utmpd* polls these files to ensure the entries for these events has been removed. Should the entries still exist in the files, *utmpd* removes them.

- *Xsun* is the X11 server for Solaris. This process is started by *xinit* through *openwin*. This process provides the graphical framework for the Window Manager to run on top of.
- *dtgreet* is the graphical login the Common Desktop Environment (CDE) provides.
- *snmpdx* is part of the Solstice package. Officially defined, *snmpdx* is the Solstice Enterprise Master Agent. This daemon listens on udp port 161, and also listens on a secondary port for snmp traps to forward.
- *ttymon* is a port monitor for terminal ports. This process is usually used in conjunction with *sac*. This facility controls TTY settings to users and services.
- *mibiisa* is the Sun RFC compliant Simple Network Management Protocol (SNMP) daemon. This daemon is fully RFC compliant, and supports additional directives, which can be found in RFC 1213.
- *snmpXdmid* is another part of the Solstice suite. This package is the DMI mapper subagent. The process is part of the Desktop Management Interface, and communicates from itself to *snmpdx*.
- *dtlogin* is another member of the CDE suite of tools distributed with Solaris. This program launches *dtgreet* when it starts. *dtlogin* also provides the vector for authentication of user credentials when logging into the system.
- *dmispd* is the DMI Service Provider. This process is the core of DMI. *dmispd* provides all the services locally for applications to use DMI, and can additionally serve and receive DMI requests remotely.
- *sendmail* is the Mail Transport Agent used on Solaris 8.
- *rpc.ttdbserverd* is part of the ToolTalk Suite. This daemon serves its database (TT\_DB) to remote clients using a Desktop Environment (normally CDE) for remote point and click usage of local resources. This service uses RPC.
- *rquotad*. This is an RPC based service that functions as part of NFS. This service enforces quotas for local file systems remotely. If a local file system is shared and remotely mounted, this daemon sends data to the remote machine in format readable by quota for the sake of quota information for users.
- **Conclusion**

So far, we've built the system, installed the Full Distribution of Solaris 8 plus OEM support, gathered information about the processes running, and discussed them.

In our next article, we'll look at how these processes are started, the significance of the startup scripts and how to customize the system to conform to our security needs.

To read **Back to the Basics: Solaris Default Processes and init.d Pt. II**, click [here](#).

[Privacy Statement](#)

Copyright 2006, SecurityFocus