AIX Booting

Article Number: 543 | Rating: Unrated | Last Updated: Tue, Apr 16, 2019 5:39 PM

Startup - Shutdown

mkitab adds record to the /etc/inittab file (-i: insert the newline anywher in the inittab file) (without the -i parameter, the line wil be appended to the end of the file)
lsitab lists record to the /etc/inittab file (lsitab -a -->lists all records of the inittab)
chitab adds record to the /etc/inittab file

uptime shows the time the server is up
last reboot shows the history when the server has been rebooted

who -b info about the last boot

/dev/ipldevice it is a hardlink to the device which holds hd5 (check major minor numbers)
ipl_varyon -i shows the state of the bootrecord

Telinit:

The telinit command sets the system at a specific run level. A run level is a software configuration that allows only a selected group of processes to exist.

who -r shows what is the runlevel of the system
cat /etc/.init.state displays the current runlevel

telinit M or ***shutdown -m*** enter to maintenance mode (single user mode)
telinit 2 return to normal mode
telinit q force the system to reread the etc/inittab file

System Management Services
Another boot option is to boot machine specific code called the System Management Services (SMS) programs. These programs are not part of AIX. This code is shipped with the hardware and is built-in to

the firmware. This can be used to examine the system configuration and set boot lists without dependency on an operating system. It is invoked using the F1 function key or the numeric 1 key.

Maintenance Mode
If your system will not boot or you have lost the root password, you will need to boot your machine using bootable media other than the hard drive (like an installation CD or NIM server). This will boot you into "maintenance" mode, which will give you "backdoor" access to your system.

BOOTING TERMINOLOGY

0. firmware bootlist:
in SMS set boot devices:disk, cd-rom, network

1. AIX bootlist:
At startup, the system searches for an AIX boot image in the boot list, a list of hdisks stored in the hardware's NVRAM.
If the system fails to boot, you can change the boot list

bootlist -m normal -o <--shows bootlist
bootlist -m normal hdisk0 hdisk1 <--sets bootlist
bootlist -m normal hdisk0 blv=hd5 pathid=0,1 hdisk1 blv=hd5 pathid=0,1,2 <--set bootlist using different paths (in given order)
bootlist -m normal en0 bserver=10.20.10.10 gateway=10.20.50.1 client=10.20.50.6 <--force the system for network boot
(bs:boot server, client:the machine what we reboot)
bootlist -m normal -ov <--it will show boot disks location codes (can be compared what sms shows during boot)

root@aix31: / # ***bootlist -m normal -ov***
'ibm,max-boot-devices' = 0x5
NVRAM variable: (boot-device=/pci@80000002000000c/pci@2/pci1069,b166@1/scsi@0/sd@4:2)
Path name: (***/pci@80000002000000c/pci@2/pci1069,b166@1/scsi@0/sd@4:4***) <--this pathname shows which blv is used
match_specific_info: ut=disk/scsi/scsd
hdisk0 blv=bos_hd5

2. boot record: (bootrec, bootstrap code)
locates the boot logical volume from the harddisk, it is the first 512 byte on the disk.

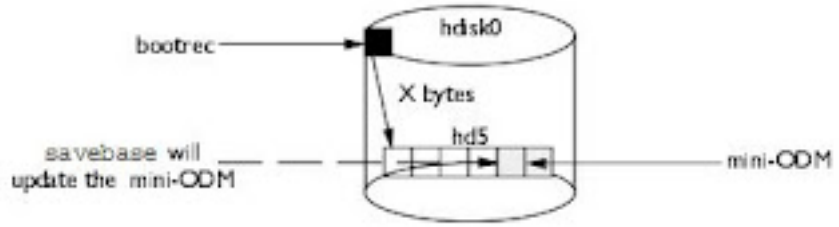
ipl_varyon -i <--shows the state of the bootrecord
bosboot -ad /dev/hdisk1 <--creates the boot record and the boot logical volume as well!!!
chpv -c hdiskX <--clears the boot record

3. boot logical volume (blv) (hd5):
Contents of the boot logical volume:
-AIX kernel (kernel is always loaded from the blv) (a copy of the kernel is under /unix)
-boot commands (cfgmgr, bootinfo)
-reduced copy of the ODM (mini-ODM)
-rc.boot: after starting the kernel it gets over the control

if blv (hd5) has to be created again: one physical partition in size, must be in rootvg and outer edge as intrapolicy.
Specify boot as logical volume type. (mklv -y hd5 -t boot -a e rootvg 1)

bosboot -ad /dev/hdisk1 <--creates the boot image on the disk
savebase -v <--updates the mini-ODM (in the boot lv) that resides on the same disk as /dev/ipldevice (-v verbose)

It may happen that 1 partition is not enough for hd5. If you add a new partition make sure the 2nd partition is next to the 1st one.
(IPL code won't be able to jump from partition 1 to 10, for example (lslv -m hd5))



The bootrec (also known as bootstrap) is read by a special part of the firmware called System ROS (- the Read Only Storage is responsibe for the initial preparation of the machine -), and it (bootrec) tells the ROS that it needs to jump X bytes into the disk platter, to read the boot logical volume, hd5. During

reading the blv, there is a mini-ODM read into the RAM. (Later, when the real rootvg fs comes online, AIX merges the data in mini-ODM with the real ODM held in /etc/objrepos.) When an LVM commands changes the mini-ODM, the command 'savebase' needed to run as well. Savebase takes a snapshot of the ODM and compresses it

---

THE AIX BOOT SEQUENCE:

1.POST
After you've turned on the power and the server is starting, the server's hardware is verified and checked for possible issues. This step is called power-on self-test (POST), it is checking the memory, keyboard, sound card, and network devices.

2. Bootstrap
After the POST process has finished, the bootstrap -or a smaller program used to load a larger program- is loaded into memory. The bootstrap then loads the Boot Logical Volume (BLV) into memory.

3. BLV
The BLV is the location that contains AIX's bootable images. Typically, the BLV can be found on the local disk of the server. The BLV contains the AIX kernel, the rc.boot file, commands required during the boot process, and a trimmed-down version of the Object Data Manager (ODM).
After the BLV is loaded, the kernel takes over the boot process.

4. The AIX kernel
The AIX kernel stored in the BLV creates the / (root), /usr, and /var file systems in RAM. These file systems as well as the kernel are stored in RAM initially during the boot process. After the file systems have been loaded into RAM, the kernel executes the init process, which was loaded out of the blv (not from the root filesystem). This init process executes rc.boot.

5. The rc.boot file
The rc.boot file has three important cases of execution during the AIX boot-up process:
-The first section of rc.boot initializes the system's hardware to prepare it for the operating system to boot.
A limited amount of devices needed to start the system are configured at this time with the Configuration Manager command cfgmgr.

-During the second phase of rc.boot, rootvg is activated and the file systems /, /usr, and /var as well as the paging space are mounted.
After these file systems have been mounted, init is replaced with init on the disk as PID 1.
All /dev files and the customized ODM files from the RAM file system are merged to disk and the RAM is cleared.

-In the third and final section of rc.boot, the actual init process is executed from disk.
When init is executed, the /etc/inittab file is read, and each item is executed and /tmp file system is now being mounted to disk.
The cfgmgr command is run again on the remaining devices that were not configured in the first section of rc.boot.

After booting, during start-up first initialisation is runnning (/etc/inittab) and after that the actual run level scripts (/etc/rc.d/rc2).
('rc' name is used in places, it is an abbreviation of 'run command')

1.SYSTEM INITIALISATION: INIT

The init process runs first. Its primary role is to start other processes listed in the /etc/inittab file. The init process rereads the /etc/inittab file every 60 seconds.

Order of the /etc/inittab entries:
1.initdefault--> init:2:initdefault (init reads the runlevel from this line, in this case 2)
2.sysinit --> brc::sysinit:/sbin/rc.boot 3 (itt calls rc.boot file with argument 3)
3.powerfailure detection (powerfail)
...
6.system resource controller (srcmstr)
7.start TCP/IP daemons (rctcpip)
8.start NFS daemons (rcnfs)
9.cron
...

format of the entries: Identifier:RunLevel:Action:Command

Identifier: uniquely identifies an object
RunLevel: the runlevel at which the enrty can be processed. (0-9)

Action: how to treat the process: respwawn, wait (wait for its termination), once, boot...
Command: A shell command to execute

The colon character (:) is used as a delimiter as well as a comment character. To comment out an inittab entry, add : at the beginning of the entry (:Identifier:RunLevel:Action:Command)

e.g..:
srcmstr:23456789:respawn:/usr/sbin/srcmstr # System Resource Controller
harc:2:wait:/usr/es/sbin/cluster/etc/harc.net # HACMP for AIX network startup
rctcpip:a:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons

---
many daemons are started from /etc/inittab:for example src, cron..and the rc.tcpip as well
rc.tcpip contains even more daemons to start: dhcpd, lpd, syslogd... <--these can be controlled usually by src
and rc.tcpip starts inetd as well which contains other daemons: ftp, telnet.. <--these should be controlled by src
---

2. Startup and Shutdown scripts:

There are more places to put these scripts:

/etc/rc.d:
-init.d: here you can put your own scripts for start/stop

-rcX.d (e.g. rc2.d): you can simply add your script to the different levels (rcX.d).
The start script must start with a capitol S and the stop scripts must start with a capitol K.
Usually the scripts in rcX.d are symbolic relative links to the actual scripts in init.d.
The scripts know whether to (S)tart or (K)ill the service by checking to see if it is running, I think.
(K and S scripts are identical (??), check Ksshd/Ssshd)

AIX does not follow the System V R4 (SVR4) run level specification. It defines run levels from 0 to 9, 0 and 1 are reserved, 2 is the default normal multiuser mode and run levels from 3 to 9 are defined by administrator.

Example start/stop script:

ls -l /etc/rc.d/rc2.d:
K99APP -> /etc/rc.d/init.d/APP
S99APP -> /etc/rc.d/init.d/APP

cat /etc/rc.d/init.d/APP:

```
#!/bin/ksh
#
# description: APP startup script
#

case "$1" in
start)
echo "start db"
su - dbuser -c /var/db/bin/start_db.ksh
sleep3
echo "start app"
su - appuser -c /app/scripts/start_app.ksh
;;

stop)
echo "stop app"
su - appuser -c /app/scripts/stop_app.ksh
echo "stop db"
su - dbuser -c /var/db/bin/stop_db.ksh
;;

*)
$ECHO "Usage: $0 {start|stop}"
exit 1
;;
esac
```


SHUTDOWN:

You can shut down the system using the shutdown command, which sends all terminals the following warning message:
shutdown: PLEASE LOG OFF. System maintenance is in progress. All the processes will be killed in one minute.

One minute after the message is displayed, the terminals are disabled, and then the system gracefully shuts down

shutdown [-options] [+time message]
-F performs a fast shutdown, sends no warning message, does not wait for applications to finish
-r reboots after shutdown, sends a warning message, and shuts down gracefully
-m shuts down into Maintenance mode, sends a warning message, and shuts down gracefully
-k sends a warning message but does not shut down the system.
(It is used to quickly determine whether it is okay to shut down the system)
-l logs the output to the file /etc/shutdown.log
-h halts the operating system completely; same as the -v flag.

The halt command shuts down the system, but it does not send a warning message. It does not wait for applications to finish processing. It is generally not used when users are logged in to the system since it does not restart the computer

The reboot command shuts down processes only. It leaves services and communications running. It does not send a warning message, and after shutdown, it checks the file systems and then restarts the system.

The fastboot command performs the same function, but it does not check the file systems when the system reboots

/etc/rc.shutdown customize shutdown sequence, it is called by the shutdown command and executed first
(for example it is useful if you need to close a databse prior to a shutdown)
If rc.shutdown fails (non-zero return code value) the shutdown sequence is terminated.

Modifying the /etc/inittab file:
(we want to add the find command to it, on run level 2, and start it again once it has finished)

1.ps -ef| grep find <--checking that no find processes are running
2.mkitab "xcmd:2:respawn:find / -type f > /dev/null 2>&1" <--adds a record named xcmd to

/etc/inittab
3.lsitab xcmd <--shows the new record
4.ps -ef | grep find <--checking the new process
5.kill <pid> <--cancel the find process
6.ps -ef | grep find <--as it is configured as respawn it will respawn
7.chitab "xcmd:2:once:find / -type f > /dev/null 2>&1" <--changes it from respawn to once (after kill <pid> it won't respawn)
(if you can change it to "off", then the line will be ignored)
8.rmitab xcmd <--deletes this record from /etc/inittab

After editing the /etc/inittab file, force the system to reread the file by using telinit q command.

Modifying the bootlist:
1. bootlist -m normal -o <--checking the devices in the bootlist)
2. bosboot -ad hdisk1 <--creates a new BLV (boot logical volume) on the disk)
3. bootlist -m normal hdisk1 hdisk0 <--change the order of the devices: (first hdisk1 and after hdisk0))

bootlist -m normal hdisk0 blv=hd5 hdisk0 blv=bos_hd5

Correct a damaged boot image:

1.lslv -m hd5 <--obtain the boot disk
2.bosboot -a -d /dev/hdiskn <--recreate the bootimage, n is the disk number of the boot logical volume)
3.shutdown -Fr <--restart the system

Changing the ipldevice:
It is Initial Program Load device. It should be a link to the device which is holding hd5 (the disk which was used for boot)

bootinfo -b <--checking disk used for boot (getconf -a | grep BOOT)
ls -l /dev/ipldevice <--check ipldevice file (find corresponding disk major, minor number)

If there is a difference between bootinfo -s and ipldevice, /dev/ipldevice should be changed:
1. rm /dev/ipldevice <--remove wrong ipldevice file

2. ln /dev/r<bootdisk> /dev/ipldevice <--create hard link for coorect rdisk (!!note the r prefix)
3. ls -i /dev/r<bootdisk> /dev/ipldevice <--check if ipldevice is correct (same inode number should be seen)

STRANGE BOSBOOT ERROR (COULD NOT CREATE BOOT IMAGE)

 # bosboot -ad hdisk0 (**bosboot -ad /dev/ipldevice** gave the same)
Error opening disk: The file access permissions do not allow the specified action.

0301-179 bosboot: The boot logical volume size must be at least 20 MB.

1. I did checks (everything looked fine):
bootlist -m normal -o
iplvaryon -i

lsvg -l rootvg
lslv hd5
lspv -M hdisk0 | head

ls -l /dev/ipldevice
ls -l /dev | grep "21, 0"

2. removed/recreated boot logical volume
 # rmlv hd5
Warning, all data contained on logical volume hd5 will be destroyed.
rmlv: Do you wish to continue? y(es) n(o)? y
0516-1246 rmlv: If hd5 is the boot logical volume, please run 'chpv -c <diskname>'
as root user to clear the boot record and avoid a potential boot
off an old boot image that may reside on the disk from which this
logical volume is moved/removed.
rmlv: Logical volume hd5 is removed.
0516-1734 rmlv: Warning, savebase failed. Please manually run 'savebase' before rebooting.

 # mklv -y hd5 -t boot -a e rootvg 1 hdisk0

hd5

I did some checks (looked fine):
 # lspv -M hdisk0 | head

3. tried bosboot again (now different error message)
bosboot -ad hdisk0 (all bosboot command gave this error)
Error opening disk: The file access permissions do not allow the specified action.

0516-602 lslv: Logical volume name not entered.
Usage: lslv [-L] [-l | -m] [-n DescriptorPV] LVname
lslv: [-L] [-n DescriptorPV] -p PVname [LVname]
Lists the characteristics of a logical volume.

0301-168 bosboot: The current boot logical volume, /dev/,
does not exist on /dev/hdisk0.

There was one bosboot command which seemed to be working (it looks it created it, with some errors):
 # bosboot -ad /dev/ipldevice -l hd5
Error opening disk: The file access permissions do not allow the specified action.
Error opening disk: The file access permissions do not allow the specified action.
Error opening disk: The file access permissions do not allow the specified action.
trustchk: Verification of attributes failed: /usr/bin/tee
: group mode
trustchk: Verification of attributes failed: /usr/sbin/bootinfo
: mode accessauths
Error opening disk: The file access permissions do not allow the specified action.
Error opening disk: The file access permissions do not allow the specified action.

bosboot: Boot image is 53276 512 byte blocks.

4. I corrected /usr/bin/tee and /usr/sbin/bootinfo (to get rid of trustchk errors)

I checked on other server these files settings (owner, permissions)

 # chmod 555 /usr/bin/tee
 # chown bin.bin /usr/bin/tee

 # ls -l /usr/sbin/bootinfo
-r-sr-xr-x 1 bin bin 14656 Sep 06 2012 /usr/sbin/bootinfo
 # chmod 550 /usr/sbin/bootinfo

5. tried again bosboot and last error did not want to disappear

 # bosboot -ad /dev/hdisk0
trustchk: Verification of attributes failed: /usr/sbin/bootinfo
: accessauths

bosboot: Boot image is 53276 512 byte blocks.

6. Found this IBM link, which gave a lot of help, but the steps there did not work
http://www-01.ibm.com/support/docview.wss?uid=isg1IV35600

 # trustchk -n /usr/sbin/bootinfo <--this is for checking if we receive the above error
trustchk: Verification of attributes failed: /usr/sbin/bootinfo
: accessauths

 # trustchk -q /usr/sbin/bootinfo 1> /tmp/file1 <-- creating a temp file

 # vi /tmp/file1 <-- vi that file and add "vios.device.manage" to accessauths part

"/tmp/file1" 21 lines, 711 characters
/usr/sbin/bootinfo:
owner = bin
group = bin
mode = 550
type = FILE
hardlinks =

symlinks =
size = 14656
cert_tag = 00af4b62b878aa47f7
signature = 4d67deebe74c64de81d84d25818e7079e60e2c9cb457c71526b8f705e5597a4b8c0cdc165050
cce6a5a4acef71ade15f0b3a878f5c5a2dfb1ae0dc8fa300ad714c25bea7f410c4e7a6ae693e10d76cac6e17
7ffb09a7
64386e01d76f119e693cedfbcaf9891c3605f69b6abf67f8dc1116bf2fa0d76e700f53fff0dadccc9f7b
hash_value = f4a586fc5048c45535493222f585601fa370afc76f83f032d42025dc97bc8235
minslabel =
maxslabel =
intlabel =
accessauths = aix.system.boot.info,vios.device.manage <-- this is the correct line, (it should look like this)
inheritprivs = PV_DAC_R,PV_DAC_W,PV_DEV_CONFIG,PV_KER_RAS
t_innateprivs = PV_MAC_,PV_MIC
inheritprivs =
authprivs =
secflags = FSF_EPS

 # trustchk -d /usr/sbin/bootinfo <--- I followed above link (have no idea about this command)
 # trustchk -a -f /tmp/file1 <--- I followed above link (have no idea about this command)

 # trustchk -n /usr/sbin/bootinfo <-- checking again if we receive any errors, and there were no more error
 # bosboot -ad /dev/hdisk0 <-- bosboot was working correctly...good job!!!

bosboot: Boot image is 53276 512 byte blocks.
