

System Boot

INDEX

Introduction	3
Step 1: Processor Dependent Code (PDC)	3
Step 2: Initial System Loader (ISL)	4
Step 3: Loading the Kernel	5
Step 4: Starting the Subsystems	7
Step 5: Console Login	10
Additional Information	12

Introduction

Once a machine powers on, the firmware controls the system until the operating system (OS) executes. The following steps need to be passed through before the system is fully operational:

- Step 1 [Processor Dependent Code \(PDC\)](#)
- Step 2 [Initial System Loader \(ISL\)](#)
- Step 3 [Loading the Kernel](#)
- Step 4 [Starting the Subsystems](#)
- Step 5 [Console Login](#)

In case of an error during the boot up, you will be able to identify the cause if you exactly know the standard boot process. This will be explained in this chapter. I recommend checking the [System Recovery Chapter](#) for non-default boot mechanisms like *single user mode*, *quorum mode* or *maintenance mode*.

Not every system boots the same. V-Class and Superdome systems do differently. Refer to the [Additional Information](#) Section for details.

Refer to the [vPars Chapter](#) to learn how a *virtual partition* is booted.

Refer to the [Itanium Chapter](#) to learn how an *Itanium system* is booted.

Step 1: Processor Dependent Code (PDC)

The *PDC* or system firmware is stored in the so called *Stable Storage*, a non-volatile piece of memory located on the processor board. It is loaded and executed when the system is powered on. It holds the addresses of primary and alternate boot device. The primary boot path will be used if the user does not interrupt by pressing any key within 10 seconds (or the ESC key at workstations):

```
Firmware Version 40.14

Duplex Console IO Dependent Code (IODC) revision 1

-----
(c) Copyright 1995-1998, Hewlett-Packard Company, All rights reserved
-----

Processor      Speed          State           CoProcessor State Cache Size
Number        MHz           Active          State           Inst     Data
-----        -----          -----          -----           -----
0            360 MHz       Active          Functional      512 KB   1 MB

Central Bus Speed (in MHz) :      82
Available Memory          :    262144 KB
Good Memory Required       :     37268 KB

Primary boot path: 0/0/2/0.0
```

```
Alternate boot path: 0/0/1/1.0
Console path: 0/0/4/0.0
Keyboard path: 0/0/4/0.0
```

Processor is starting autoboot process.

To discontinue, press any key within 10 seconds.

<key pressed>

Boot terminated.

```
----- Main Menu -----
Command                                     Description
-----
B0ot [PRI|ALT|<path>]                   Boot from specified path
PAth [PRI|ALT] [<path>]                  Display or modify a path
SEArch [DIsplay|IPL] [<path>]             Search for boot devices

COnfiguration menu                      Displays or sets boot values
INFormation menu                         Displays hardware information
SERvice menu                            Displays service commands

DIggest
HELP [<menu>|<command>]                 Redisplay the current menu
RESET                                    Display help for menu or command
                                         Restart the system
-----
Main Menu: Enter command or menu >
```

NOTE: The boot menu may look different depending on the system that is being booted.

See the `pdc` man page for further details.

Step 2: Initial System Loader (ISL)

The PDC now looks for a LIF header on the device and - if it finds one - it looks in the LIF header for where the ISL starts. A typical LIF header residing at the start of a **LVM** boot disk looks like this:

```
# lifls -l /dev/rdsk/c1t6d0
volume ISL10 data size 7984 directory size 8
filename    type   start    size   implement   created
=====
ISL        -12800  584     306      0   00/11/08 20:49:59
HPUX       -12928  896     848      0   00/11/08 20:50:00
LABEL      BIN     1744     8       0   01/05/31 06:54:08
AUTO       -12289  1752     1       0   01/05/31 06:54:10
```

The values represent 256 Byte units, so the ISL starts e.g. at offset **146K**. All LVM boot disks (created with `pvcreate -B`) reserve space between their PVRA and VGRA to hold a BDRA and the LIF files. Their LVM header size is always **2912K**. In addition to the files above there may be additional files, e.g. for the Offline Diagnostics. A PAD file may be used to fill (pad) unneeded space with zeros.

Refer to the [LVM Chapter](#) to get an explanation of the terms above.

Now the *initial program loader* (IPL) loads the ISL into memory from the boot device and executes it. It passes a flag to it that indicates whether to run interactively or to autoboot. If the ISL is interactive then it gives the `ISL>` prompt and waits for user input before proceeding:

```
Main Menu: Enter command or menu > bo
Interact with IPL (Y, N, or Cancel)?> y

Booting...
Boot IO Dependent Code (IODC) revision 1

HARD Booted.

ISL Revision A.00.43 Apr 12, 2000

ISL> help

HELP          Help Facility
LS             List ISL utilities
AUTOBOOT      Set or clear autoboot flag in stable storage
AUTOSearch    Set or clear autosearch flag in stable storage
PRIMPATH      Modify primary boot path in stable storage
ALTPATH       Modify alternate boot path in stable storage
CONSPATH      Modify system console path in stable storage
DISPLAY       Display boot and console paths in stable storage
LSAUTOFL      List contents of autoboot file
FASTSIZE      Sets or displays FASTSIZE
800SUPPORT   Boots the s800 Support Kernel from the boot device
700SUPPORT   Boot the s700 Support Kernel from the boot device
READNVM      Displays contents of one word of NVM
READSS        Displays contents of one word of stable storage
LSBATCH       List contents of batch file
BATCH         Execute commands in batch file
LSEST         List contents of EST (Extended Self Test) file
EST           Execute commands in EST (Extended Self Test) file

Enter 'LS' to see a list of the ISL utilities.
```

ISL>

NOTE:

There is no ISL for V-Class Systems. It is part of PDC and called open boot prompt (OBP).

If the ISL is not interactive then it looks for the AUTO file on the LIF volume of the boot device to determine what to run next.

See `isl` man page for further details.

Step 3: Loading the Kernel

The `hpx` program (also known as the *secondary loader*) figures out what HP-UX kernel to load and what arguments to pass to it. The AUTO file or user input supply the loader's arguments. You can display the content of the AUTO file within ISL:

```
ISL> lsa
Auto-execute file contains:
hpx
```

or from the command line of a running system:

```
# lifcp /dev/rdsk/c1t6d0:AUTO -  
hpxx
```

You can also pass arguments to hpxx in order to specify the disk and section of the disk to boot from, the kernel file and whether to boot to single user, maintenance or quorum mode. The Syntax is as follows:

```
hpxx (<HW path of boot disk>;<section>)<path to kernel> <options>
```

simply using hpxx is equal to:

```
hpxx ( ;0)/stand/vmunix
```

The possible options are:

-is single user mode boot

Activate vg00, mount only / and /stand, do not go through the startup script.
So e.g. the network is not started and other users cannot login.

-lq quorum mode boot

Same as single user mode but the root VG (vg00) gets activated even if the LVM quorum is not present, i.e only 50% or less of the disks are available. This option can be used if a mirrored root disk is defect.

-lm maintenance mode boot

vg00 will not be activated. The root FS is mounted using the auxiliary device file /dev/root. In this mode you are able to export vg00 and perform LVM low level troubleshooting.

You may also combine the options above. In order to boot without quorum check into single user mode type:

```
ISL> hpxx -is -lq
```

hpxx provides some other useful features like:

```
ISL> hpxx ll
```

shows the content of the boot filesystem /stand.
This is useful if the kernel cannot be found

```
ISL> hpxx show autofile
```

The content of the LIF file AUTO is shown. This is identical to the ISL utility lsa (see above)

```
ISL> hpxx set autofile <string>
```

Sets the content of the LIF file AUTO to whatever you specify in <string>.

For additional arguments of hpxx, see hpxx man page.

NOTE: Things are different on a virtual partition (vPars) system. You do not load the kernel directly, but through the vPars monitor vpmon. Refer to the [vPars Chapter](#) for details.

Now the secondary loader relocates itself to the end of the initial memory module (IMM),

loads the kernel and starts running it. In the next steps the kernel initializes memory, IO and forks off system daemons.

1. Kernel initialization (real mode):

Initialize all of the memory, read /stand/ioconfig and /stand/rootconf files using the hpx loader's system calls, initialize all modules (1st level I/O configuration), allocate equivalently-mapped data structures, PDIR and hash table, optimize assembly, craft process 0, go virtual.

2. Kernel initialization (virtual mode):

Start the clock, start up the other processors, finish the I/O configuration (2nd level), initialize subsystems, initialize LVM/swap/dump, mount root file system read-only, fork() off system daemons.

3. fork() off /sbin/pre_init_rc and mount root file system read-write afterwards.

The console output is as follows:

```
ISL> hpx

Booting...
Boot IO Dependent Code (IODC) revision 1

HARD Booted.

ISL Revision A.00.43 Apr 12, 2000

ISL booting hpx

Boot
: disk(0/0/2/0.0.0.0.0.0;0)/stand/vmunix
9310208 + 1843200 + 1733192 start 0x1f0ae8

alloc_pdc_pages: Relocating PDC from 0xf0f000000 to 0x1fb01000.
gate64: sysvec_vaddr = 0xc0002000 for 2 pages
NOTICE: autofs_link(): File system was registered at index 3.
NOTICE: cachefs_link(): File system was registered at index 5.
NOTICE: nfs3_link(): File system was registered at index 6.

System Console is on the Built-In Serial Interface
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
Swap device table: (start & size given in 512-byte blocks)
    entry 0 - major is 64, minor is 0x2; start = 0, size = 1048576
Starting the STREAMS daemons-phase 1
Checking root file system.
file system is clean - log replay is not required
Root check done.

Memory Information:
    physical page size = 4096 bytes, logical page size = 4096 bytes
    Physical: 524288 Kbytes, lockable: 363332 Kbytes, available: 417652 Kbytes
```

Step 4: Starting the Subsystems

The /etc/init process (pid 1) gets forked off which, depending on the passed init state,

starts working through `/etc/inittab` or launches a shell in the case of a single user or LVM maintenance boot.

The following is the default inittab for HP-UX 11.11:

```
# cat /etc/inittab

#ups::respawn:rtprio 0 /usr/lbin/ups_mond -f /etc/ups_conf

init:3:initdefault:
ioin::sysinit:/sbin/ioinitrc >/dev/console 2>&1
tape::sysinit:/sbin/mtinit > /dev/console 2>&1
muxi::sysinit:/sbin/dasetup </dev/console >/dev/console 2>&1 # mux init
stty::sysinit:/sbin/stty 9600 clocal icanon echo opost onlcr ixon icrnl ignpar
</dev/sttty
brc1::bootwait:/sbin/bcheckrc </dev/console >/dev/console 2>&1 # fsck, etc.
link::wait:/sbin/sh -c "/sbin/rm -f /dev/syscon; \
                               /sbin/ln /dev/systty /dev/syscon" >/dev/console 2>&1
cprt::bootwait:/sbin/cat /etc/copyright >/dev/syscon          # legal req
sqnc::wait:/sbin/rc </dev/console >/dev/console 2>&1          # system init
#powf::powerwait:/sbin/powerfail >/dev/console 2>&1          # powerfail
cons:123456:respawn:/usr/sbin/getty console console          # system console
#ttpl:234:respawn:/usr/sbin/getty -h tty0p1 9600
#ttpl:234:respawn:/usr/sbin/getty -h tty0p2 9600
#ttpl:234:respawn:/usr/sbin/getty -h tty0p3 9600
#ttpl:234:respawn:/usr/sbin/getty -h tty0p4 9600
#ttpl:234:respawn:/usr/sbin/getty -h tty0p5 9600
krsd:123456:respawn:/sbin/krsd -i
sfd:123456:respawn:/sbin/sfd
```

The inittab file is composed of entries that are position-dependent and have the following format:

id:rstate:action:process

The entry fields are:

id	A one- to four-character value used to uniquely identify an entry.
rstate	defines the run level in which this entry is to be processed. A process can be assigned to one ore more run levels. No enjtry in this fields means all run level.
action	A keyword in this field tells boot init how to treat the process specified in the process field. Here are some of the actions that can be specified:
boot	Process the entry only at boot init's boot-time read of the inittab file. Boot init starts the process, does not wait for its termination, and when it dies, does not restart the process.
bootwait	Process the entry only at boot init's boot-time read of the inittab file. Boot init starts the process, waits for its termination, and, when it dies, does not restart the process.
initdefault	the default run level

respawn If the process does not exist, start the process; do not wait for its termination (continue scanning the inittab file). When it dies, restart the process. If the process currently exists, do nothing and continue scanning the inittab file.

process This is the shell command to be executed.

The processes that are started by default are:

ioinitrc, mtinit, dasetup and stty

Initialization of IO subsystem tape devices multiplexer and console respectively.

bcheckrc

/sbin/bcheckrc activates the LVM volume groups and checks the filesystems, Here's the console output:

```
/sbin/bcheckrc:
Checking for LVM volume groups and Activating (if any exist)
Volume group "/dev/vg00" has been successfully changed.
vxfs fsck: sanity check: root file system OK (mounted read/write)
Checking hfs file systems
/sbin/fsclean: /dev/vg00/lvol1 (mounted) ok
HFS file systems are OK, not running fsck
Checking vxfs file systems
/dev/vg00/lvol8 :
vxfs fsck: sanity check: /dev/vg00/lvol8 OK
/dev/vg00/lvol9 :
vxfs fsck: sanity check: /dev/vg00/lvol9 OK
/dev/vg00/lvol3 :
vxfs fsck: sanity check: root file system OK (mounted read/write)
/dev/vg00/lvol4 :
vxfs fsck: sanity check: /dev/vg00/lvol4 OK
/dev/vg00/lvol5 :
vxfs fsck: sanity check: /dev/vg00/lvol5 OK
/dev/vg00/lvol6 :
vxfs fsck: sanity check: /dev/vg00/lvol6 OK
/dev/vg00/lvol7 :
vxfs fsck: sanity check: /dev/vg00/lvol7 OK
```

rc

The run commands are explained in detail in the [System Startup Chapter](#). The console output is like the following:

```
HP-UX Start-up in progress

Mount file systems ..... OK
Setting hostname ..... OK
Set privilege group ..... N/A
Display date ..... N/A
Save system core image if needed ..... N/A
Enable auxiliary swap space ..... OK
Start syncer daemon ..... OK
Configure LAN interfaces ..... OK
Start Software Distributor agent daemon ..... OK
Configuring all unconfigured software filesets ..... OK
Recover editor crash files ..... OK
Clean UUCP ..... OK
```

List and/or clear temporary files	OK
Clean up old log files	OK
Start system message logging daemon	OK
Start pty allocator daemon	OK
Start network tracing and logging daemon	OK
Configure HP Ethernet interfaces	OK
Configure LAN interfaces	OK
Start name server daemon	N/A
Start NFS core subsystem	OK
Start NIS server subsystem	OK
Start NIS client subsystem	OK
Start NFS client subsystem	OK
Start multicast routing daemon	N/A
Start Internet services daemon	OK
Start dynamic routing daemon	N/A
Start router discover protocol daemon	N/A
Start RARP protocol daemon	N/A
Start remote system status daemon	N/A
Starting mail daemon	OK
Starting outbound connection daemons for DDFA software	N/A
Start SNMP Master Network Management daemon	OK
Start OSPF MIB Network Management subAgent	N/A
Start SNMP HP-UNIX Network Management subAgent	OK
Start SNMP MIB-2 Network Management subAgent	OK
Start DCE daemons	N/A
Start DFS daemons	N/A
Start NCS broker daemons	OK
Start Gradient License Server Daemon	N/A
Start remote boot daemon	N/A
Starting X Font Server at TCP port 7000	N/A
Start vt daemon	OK
Start time synchronization	N/A
Start accounting	N/A
Starting disk array monitor daemons	OK
Start print spooler	N/A
Starting HP Distributed Print Service	N/A
Start clock daemon	OK
Start environment monitoring daemon	OK
Start auditing subsystem	N/A
Start audio server daemon	N/A
SAM Single point administration configuration	N/A
Installing software to diskless nodes	OK
Schreibe Datum	OK
Start NFS server subsystem	OK
Start CDE login server	OK

The system is ready.

For more details refer to the man pages of `init(1M)` and `inittab(4)`.

Step 5: Console Login

The boot process finishes with the start of the getty process that provides the login:

```
Console Login: root
Password: *****

Please wait...checking for disk quotas
(c)Copyright 1983-1996 Hewlett-Packard Co., All Rights Reserved.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
(c)Copyright 1980, 1984, 1986 Novell, Inc.
(c)Copyright 1986-1992 Sun Microsystems, Inc.
(c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
(c)Copyright 1989-1993 The Open Software Foundation, Inc.
```

(c)Copyright 1986 Digital Equipment Corp.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
(c)Copyright 1988 Carnegie Mellon University

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in sub-paragraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).
You have mail.

Value of TERM has been set to "70094".
WARNING: YOU ARE SUPERUSER !!

Additional Information

Manual Pages

`boot(1M)`, `pdc(1M)`, `isl(1M)`, `hpx(1M)`, `init(1M)`, `inittab(4)`

Additional information about the boot process can be found in the [System Recovery Chapter](#).

V-Class boot process is described in the *V-Class Operator's Guide*.

Superdome boot process is described in the *HP System Partitions Guide*.

Both documents can be found at <http://docs.hp.com/hpx/hw>.

VxVM Maintenance Mode Boot white paper

http://docs.hp.com/hpx/onlinedocs/os/11iV1.5/vxvm_mmb.html

Refer to the [vPars Chapter](#) to learn how a *virtual partition* is booted.

Refer to the [Itanium Chapter](#) to learn how an *Itanium system* is booted.