netdirect
open solutions

SERVER ISSUES KEEPING YOU UP?

CHOOSE NET DIRECT

Linux Boot Process and Startup

Pulling oneself up by one's own bootstraps

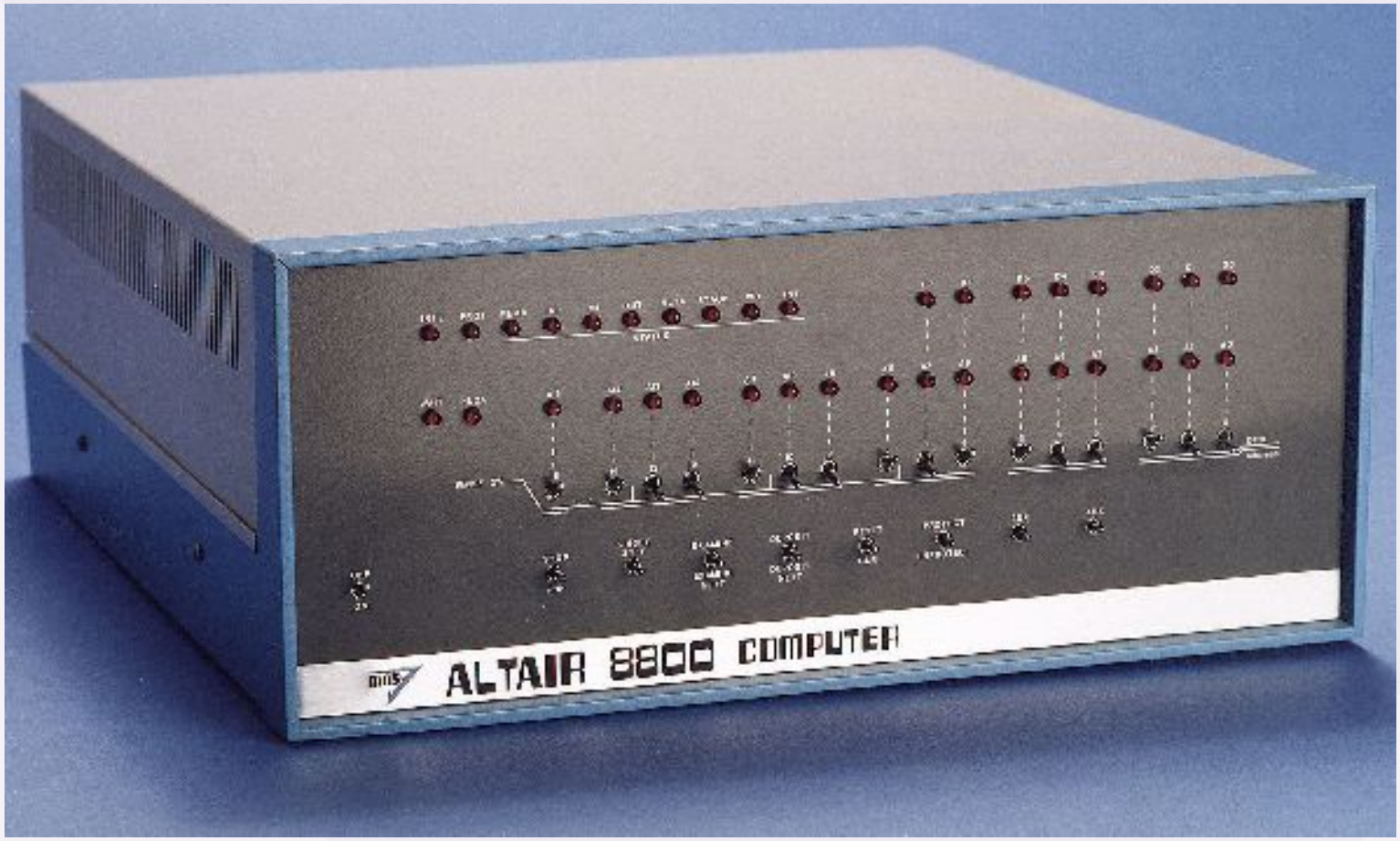
Your Presenter

- John Van Ostrand
- Linux user since 1994, 1995
- UNIX user since 1992
- Owner Net Direct Inc.
- Organizer Ontario Linux Fest
- Avid open source enthusiast

Introduction

- Bootstrapping (booting) is the process of bringing a computer from a power-on state to fully functional.
- Involves locating and running simpler programs that in-turn load more complicated ones until the system is “smart” enough to do something interesting.

Historical Booting



Intel 8080, 2Mhz, 256 Bytes

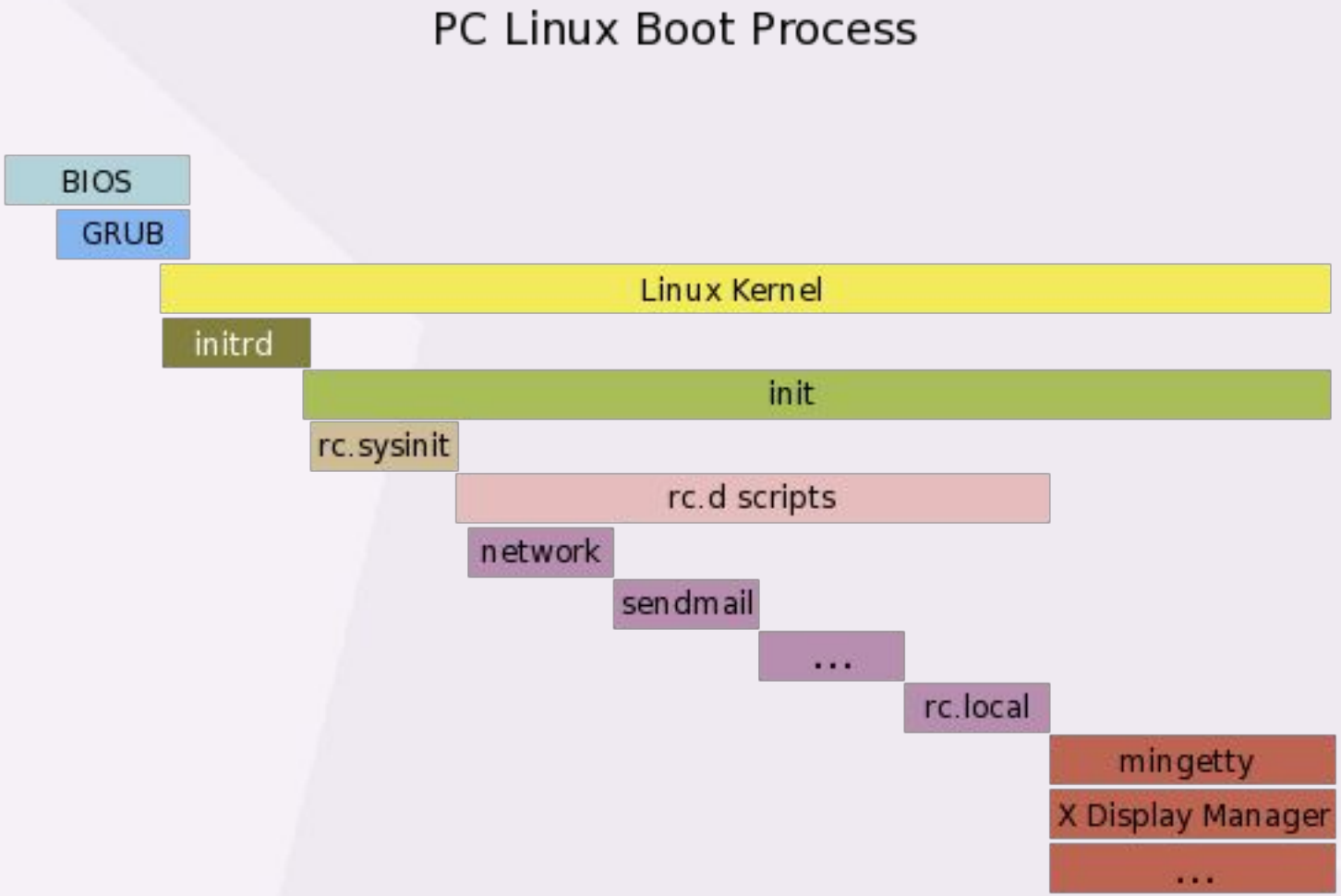
No Software, $621 assembled

Input method: toggle switches

Boot process Overview

- BIOS runs when computer is turned on
- BIOS loads and runs boot loader program from a device (or network)
- Boot loader loads Linux kernel and possibly an initial ram disk into memory and runs kernel
- Linux kernel initializes, mounts initial ram disk, and runs “init” from ram disk
- init loads drivers, mounts file systems and runs “init” from root file system
- init runs rc scripts and other programs

Boot Process Graphically



BIOS

- Firmware (software on a chip) that initializes and interfaces with devices at a basic level.
- Provided by the various devices generally the motherboard and adapter cards.
- Motherboard provides keyboard, USB, floppy, and SATA and/or ATA.
- Display, SCSI, RAID, USB and network adapters are common providers of BIOS code.
- Provides a consistent interface to hardware for boot programs and operating systems

Architecture Dependent

- Most PCs use “BIOS” (Basic Input/Output System)
- PowerPC (Macs, IBM System p) and SPARC (Sun) use “Open Firmware”
- Intel Macs, Itanium and some x86_64 use “Extensible Firmware Interface” (EFI)
- Coreboot (formerly Linux BIOS project) can be used in a selection of x86 motherboards

What the BIOS Does

- Checks and initializes the hardware
- Checks it's configuration for device information and boot device order
- Loads the boot loader from the boot sector (first 446 bytes of hard disk or floppy) and runs it
- Continues to provide I/O functions for boot loader and for some really old O/Ses

Grub Boot Loader

- Stages 1, 1.5 and 2
- Stage 1 loads stage 1.5 found in space between the first sector and the first partition
- Stage 1.5 can read ext2 file system. It reads the /boot filesystem, then loads and runs Stage 2.
- Stage 2 reads menu.lst (a.k.a. grub.conf) and prompts user.
- Stage 2 loads Kernel and initial ram disk
- All grub stages use BIOS to access disk.

LILO Boot Loader

- Two stages
- Stage 1 is loaded and run by BIOS it loads stage 2 using a sector map written at install time
- Stage 2 loads additional data and prompts user.
- Stage 2 then loads kernel and initial ram disk using a sector map written at install
- All LILO stages use BIOS to access disk

Here comes Linux

- Bootloader loads and runs kernel with optional initial ram disk (initrd)
- Kernel is “told” by bootloader of the ram disk.
- Kernel initializes
- Kernel runs “init” from RAM disk. Init is a script that loads drivers and mounts root filesystem.
- Init loads additional drivers (e.g. storage controller, LVM, file system, software RAID, etc.)

Initial RAM disk

- Solves chicken-and-egg problem caused by modular kernel
  - Kernel needs to access storage for storage drivers but needs the storage driver to access storage
- Provides initial drivers for kernel to continue to boot
- A CPIO archive (prior to 2.6 it was an ext2 disk image)
- Kernel runs “init” from the initrd filesystem
- “init” is a script that loads drivers and mounts file systems

Working with initrd

- Unpack an initrd to see what's inside:
  - mkdir /tmp/ir
  - cd /tmp/ir
  - zcat /boot/initrd-2.6.22.5-31 | cpio -ivdum
- Create an initrd
  - mkinitrd /boot/initrd-2.6.22.5-31 2.6.22.5-31
  - Varies by distro
- Or manually (after altering extracted image)
  - find . | cpio -ovH newc | gzip > /boot/initrd-2.6.22.5-31

The mother of all processes

- Kernel runs “init” as the first process, PID 1
- Init reads /etc/inittab and runs programs configured within
- All processes have 1 as the top-level parent PID
- Can be overridden using “init=” on the kernel command line

inittab file

- Line format is:
  - id:runlevels:action:command
- ID is just a unique ID
- Run levels are semi-arbitrary states: 0 - 6
- Action describe how to run processes:
  - initdefault, sysinit, bootwait, wait, respawn, off
- Command is what is run

Run levels

- Run levels are semi-arbitrary states of the operating system.
- Generally though they are commonly defined as
  - 0 = shutdown
  - 1 = single user mode
  - 2 = multi-user mode, no network
  - 3 = multi-user mode with network
  - 4 = undefined (user definable)
  - 5 = graphical user interface
  - 6 = reboot

Run Levels

- initially set in one of two ways:
  - on the kernel command line (just specify the number, or 's' for 0)
  - from inittab file (the line with “initdefault” action)
- Set manually on the command line
  - init #
- Current (and last) run level can be seen with:
  - who -r

Example inittab entries

- Set default run level to “5”, GUI
  - id:5:initdefault
- Run “boot” script and wait until completed
  - si::bootwait:/etc/init.d/boot
- Run”rc” script for run level 5
  - l5:5:wait:/etc/init.d/rc 5
- Run console login and re-run when it ends
  - 1:2345:respawn:/sbin/mingetty -noclear tty1

boot or sysinit script

- Name depends on the distro
- First script run (after initial ram disk) at boot
- Does basic O/S setup like:
  - set console fonts, set hostname, mounts pseudo file systems (/proc, /sys, /dev, /dev/pts), shows splash screen or graphical boot, re-mounts root file system as read-write.
  - Prompts user with single user mode if problems arise.

init Scripts

- Two main types: SYSV or BSD style init scripts
- BSD uses a single (or only a few) monolithic RC scripts to start everything
- SYSV uses a script for every “package” that needs boot time startup.
- BSD tends to be faster, but harder to configure
- SYSV is very package friendly, more convenient to start and stop individual packages

BSD init Scripts

- init runs the “rc” script as configured in inittab
- rc script runs a few child scripts, rc.network, rc.local
- If a new package is installed one of these scripts need to be hand configured
- Commonly used on slackware

SYSV init Script

- init runs “rc” script passing the run level
- rc then looks in /etc/rc.d/rc#.d (#=runlevel)
- Scripts starting with K are run with “stop” as an argument. Scripts starting with S are run with “start” as the argument
- Scripts are run in alphabetical order.
- These K and S scripts are symlinks to scripts in /etc/init.d/

Example rc5.d directory

- Order is important, e.g. the network needs to be up before network services:
  - S06cpuspeed (->../init.d/cpuspeed)
  - S10network (-> ../init.d/network)
  - S12syslog (-> ../init.d/syslog)
  - S55sshd (-> ../init.d/sshd)
  - S85httpd (-> ../init.d/httpd)
  - S99local (-> ../rc.local)
- “S” and “K” files are symlinks to the installed scripts. It allows services to be enabled and disabled. without deleting the script.

Simple init script

```
#!/bin/bash

case   $1   in
 start  # Start something
   /usr/sbin/mydaemon
   ;;
 stop  ) # Stop something
   killproc mydaemon

   ;;
*)   # Invalid argument
   echo   Invalid argument
   ;;
esac
```

Using chkconfig

- Chkconfig sets up symlinks in /etc/rc?.d directory (? is runlevel)
- Actual scripts are often in /etc/init.d/
- chkconfig also works on “inetd” based services (more on that later)
- chkconfig reads the script and looks for “chkconfig:” and “description” comments

Setting up an initscript for chkconfig

- Add lines like this:
  - # chkconfig: 345 95 05
  - # description: A description of my init script
- The “chkconfig” line has these options:
  - 345: default run levels to install to
  - 95: the startup sequence number. It becomes part of the startup script filename e.g. S95mydaemon
  - 05: the shutdown sequence number. It becomes part of the symlink name, e.g. K05mydaemon

Using chkconfig

- To add a script and set at the default run level
  - Place the script in /etc/init.d and chmod a+rx
  - run “chkconfig –add mydaemon
- To see at which run levels a script will run:
  - chkconfig –list mydaemon
- To prevent a script from running at boot:
  - chkconifg mydaemon off
- And to set it to start:
  - chkconfig mydaemon on

insserv

- Alternative to chkconfig (distro dependent)
- Similar to chkconfig in that it reads the script for additional info
  - Provides: mydaemon
  - Required-Start: $network $remote_fs
  - Required-Stop: $network $remote_fs
  - Default-Start: 3 4 5
  - Default-Stop: 0 1 2 6
  - Description: A description of the script
- insserv determines start order based on “Required-*” lines.

Using insserv

- To add an init script or set one to start at boot:
  - insserv -d mydaemon
- To prevent an init script from running at boot:
  - insserv -r mydaemon

No chkconfig or insserv

- rc?.d directories contain symlinks to init.d directory
- Create the symlinks manually:
  - cd /etc/rc5.d
  - ln -s ../init.d/httpd S99httpd
  - ln -s ../init.d/httpd K01httpd

Starting/Stopping Services Manually

- Some distros (RedHat Fedora) have a “service” command
  - service mydaemon start
  - service mydaemon stop
- Others (SUSE, Debian) you have to run the script:
  - /etc/init.d/mydaemon start
  - /etc/init.d/mydaemon stop
-

Debugging startup scripts

- If automatic startup is failing try manual
- Check /var/log/messages
- Run script manually in debug mode:
  - sh -x /etc/init.d/mydaemon start
- Check to see if dependencies are started
- Put “echo” statements in the script to see what's happening if problem only occurs at boot time

Other Startups

- init triggers a cascade of programs some interactive and some otherwise significant
  - xinetd
  - console/terminal login
  - GUI login
  - udev

xinetd

- The “superdaemon”
- Started by rc scripts
- Listens to network ports for various services. Runs the actual service command when a client connects.
- Originally intended to reduce overhead by replacing many big but seldom used daemons with one smaller daemon.
- Sometimes not used at all.

xinetd config

- Configured in /etc/xinetd.conf and by the files in /etc/xinetd.d, e.g.:

```
service tftp {
        socket type = dgram
        protocol = udp
        wait = yes
        user = root
        server = /usr/sbin/in.tftpd
        server_args = -s /tftpboot
        disable = no
}
```

User profiles

- Setup or start things at login time
- System-wide configuration in:
  - /etc/bashrc (or /etc/bash.bashrc)
  - /etc/profile or /etc/profile.d/*
- Per-user configuration in
  - ~user/.profile or ~user/.bash_profile
  - ~user/.bashrc
- New user defaults set in:
  - /etc/skel/.profile or /etc/skel/.bash_profile
  - /etc/skel/.bashrc

Graphical Application Startup

- For “normal” graphical logins, configure .xsession to start apps at login
- Server wide changes can be made to /etc/X11/xdm
- For “startx” sessions use .initrc

udev

- Alternative to static device entries in /dev
- Devices are created as they are discovered
- Consistent device names (USB devices)
- Customized ownership and perms
- Can run programs when devices are configured
- Configured in /etc/udev/rules.d
- Too complicated for this presentation

Contact Information

- http://www.netdirect.ca
- 866-883-1172
- info@netdirect.ca
- Questions??