The Linux IPL Procedure

SHARE 2008 San Jose – Session 9274

Edmund R. MacKenty
Rocket Software, Inc.

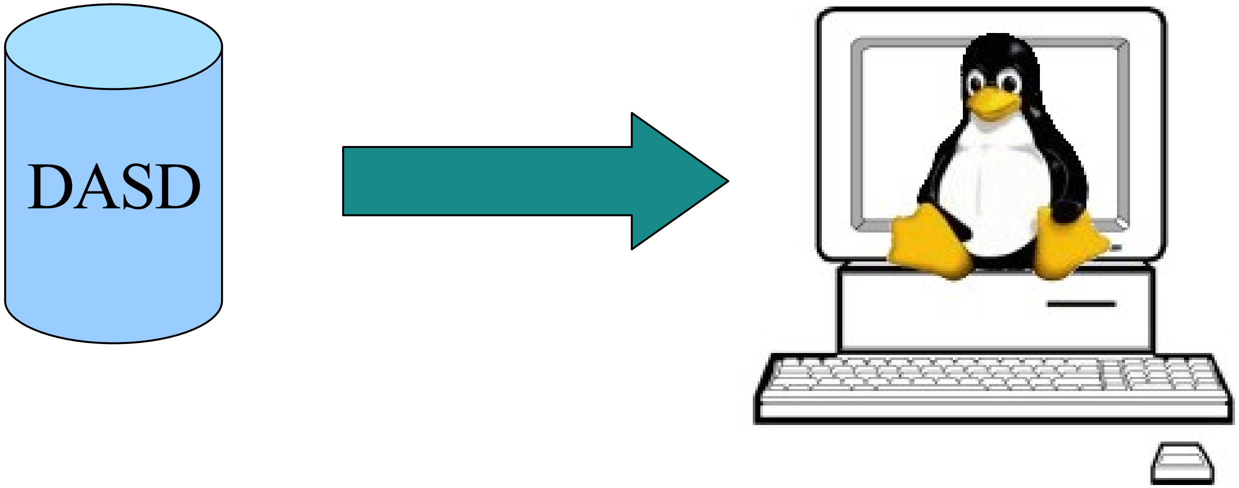
Your primary source for enterprise software

Rocket®

MAINSTAR

Purpose

- De-mystify the Linux boot sequence
- Explain what happens each step of the way
- Describe why each step exists
- Tell you how to learn more



General Design Principles

- Flexibility: uses not thought of by designers
- Extensibility: accommodate specific end-user needs
- Reuse-ability: of code and user data
- Controllability: higher-level code can drive it
- Portability: can operate in different environments
- Simplicity: easy to understand, use; limited side-effects

Overview

- Boot loader
- Kernel
- Initial RAM disk
- Init process
- Runtime configuration scripts
- User login

Concepts

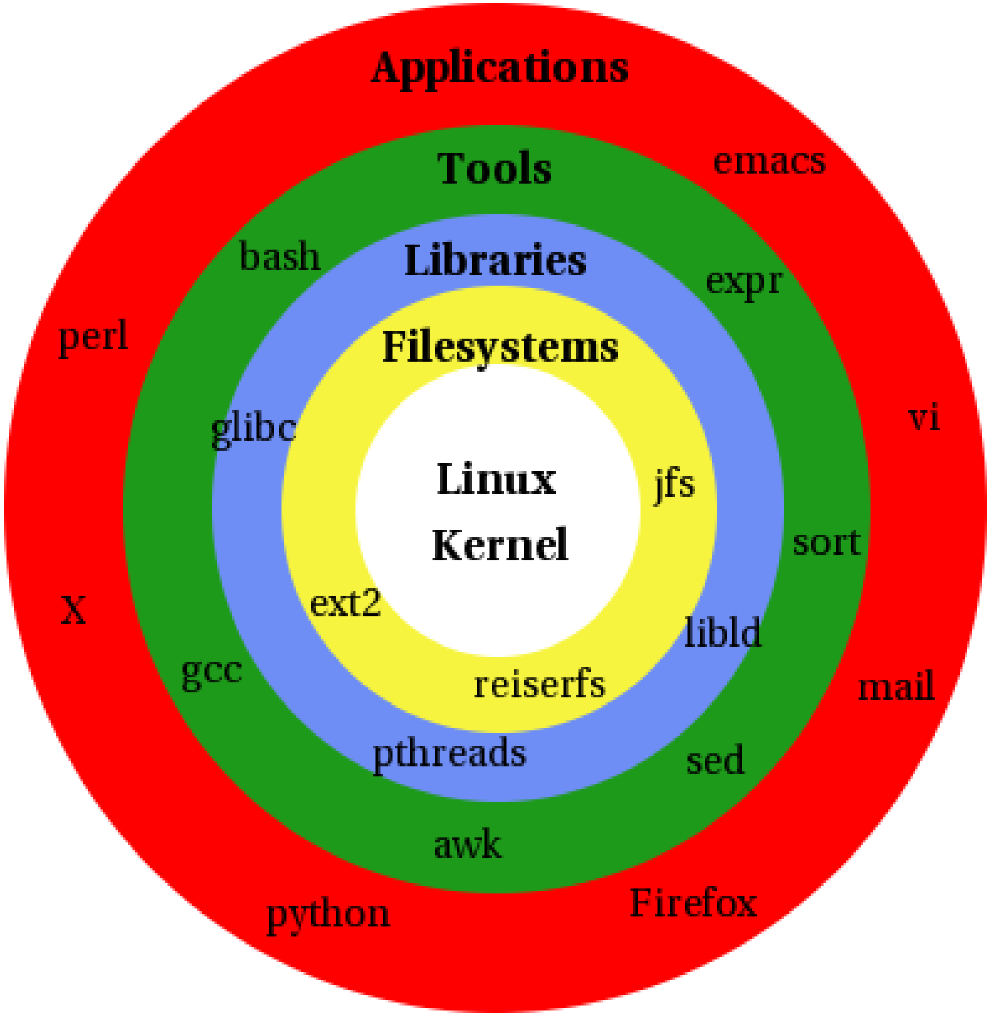
- The Kernel
- Device drivers
- Kernel modules
- Filesystems
- Mounting a filesystem
- Processes
- The onion
- The two trees
- Run Levels
- The online manual

Manual Pages

- Online manual is a good source of information
- References to manpages use the form: *page*(*section*)
  - Section 1: User Commands
  - Section 2: System Calls
  - Section 3: Library Functions
  - Section 4: Special Files
  - Section 5: File Formats
  - Section 6: Games
  - Section 7: Conventions and Miscellany
  - Section 8: Administrative Commands
- To learn about `init(8)`, use the command: `man 8 init`
- Use info(1) for more information about some commands
- The apropos(1) and whatis(1) commands do searches
- Different distros have different manpages available

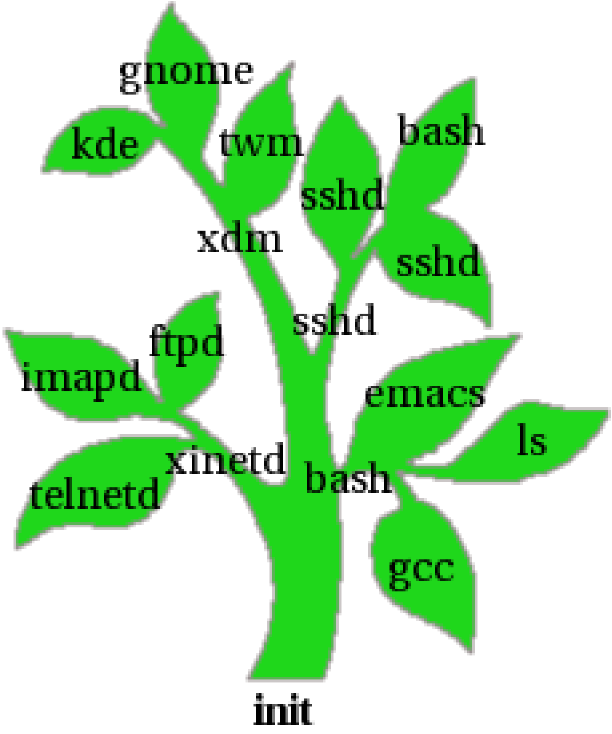
Linux: The Onion

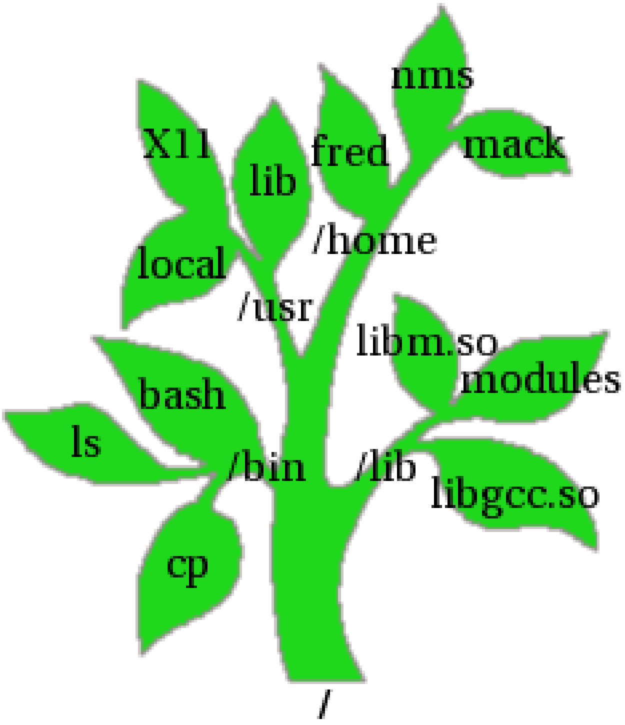
- Linux consists of many layers surrounding a kernel



Linux: The Two Trees

- Linux consists of two trees: Processes and Files
- Processes inherit properties from their parent
- Files reside within their parent directory



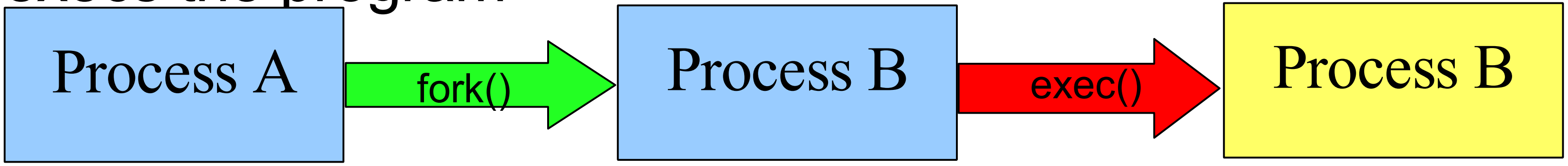


Structure of the Kernel

- The Linux kernel is not monolithic
- All device drivers and many sub-components may be built as **modules**, which can be loaded or unloaded as needed.
- This permits one kernel to run efficiently on lots of different hardware.
- The kernel build process is *amazingly* configurable.
- Some core components must be compiled in:
  - Memory management
  - Virtual filesystem layer
  - Process scheduler
  - Multi-processor support
  - TCP/IP networking (if used)
- Examples of dynamically-loaded modules:
  - Filesystems: ext3, reiserfs, jfs
  - Support for specific hardware: SCSI, DASD, USB, Crypto
  - Network drivers

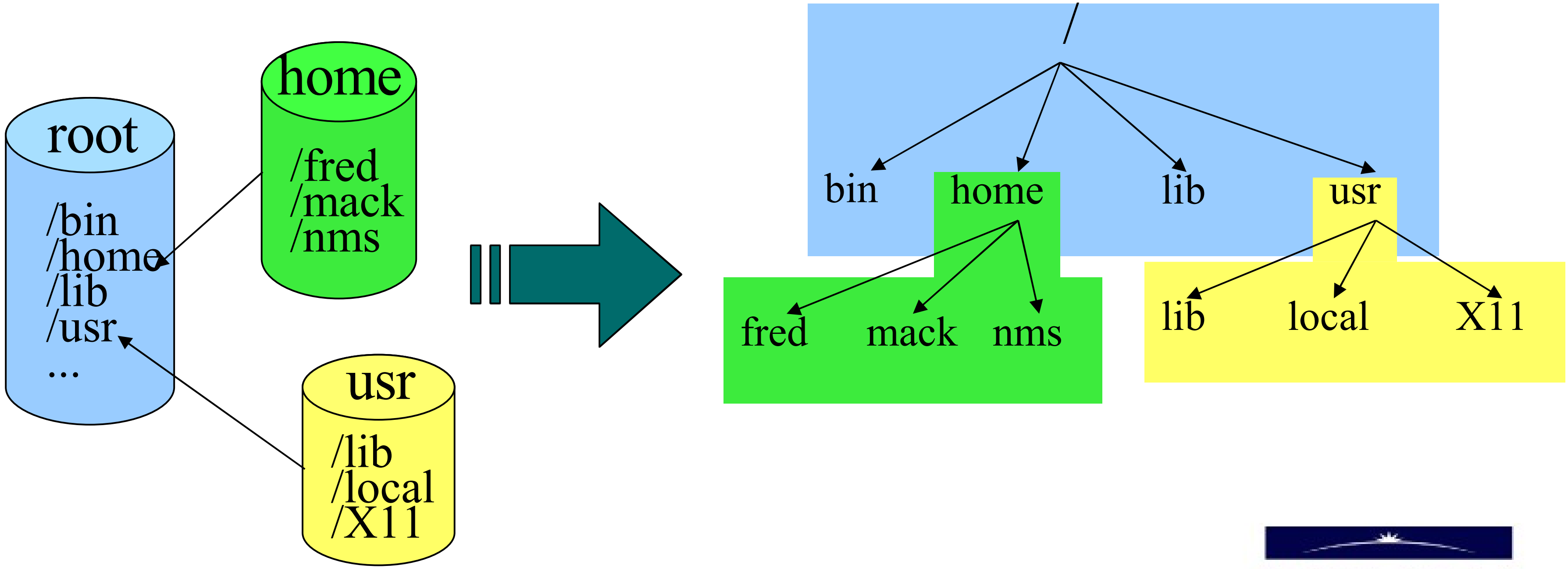
Processes

- A unit of execution scheduled by the kernel
- Each process runs in its own address space
- Fork: creates a new, child, process
  - Inherits code and data segments
  - Gets copies of all open files, sockets, etc.
  - Process execution returns from fork() call
- Exec: Loads a new program into a process
  - All open files are closed
  - New code and data segments are allocated
  - Process execution continues at entry point of new program
- “Running a program” means a process forks and the child execs the program



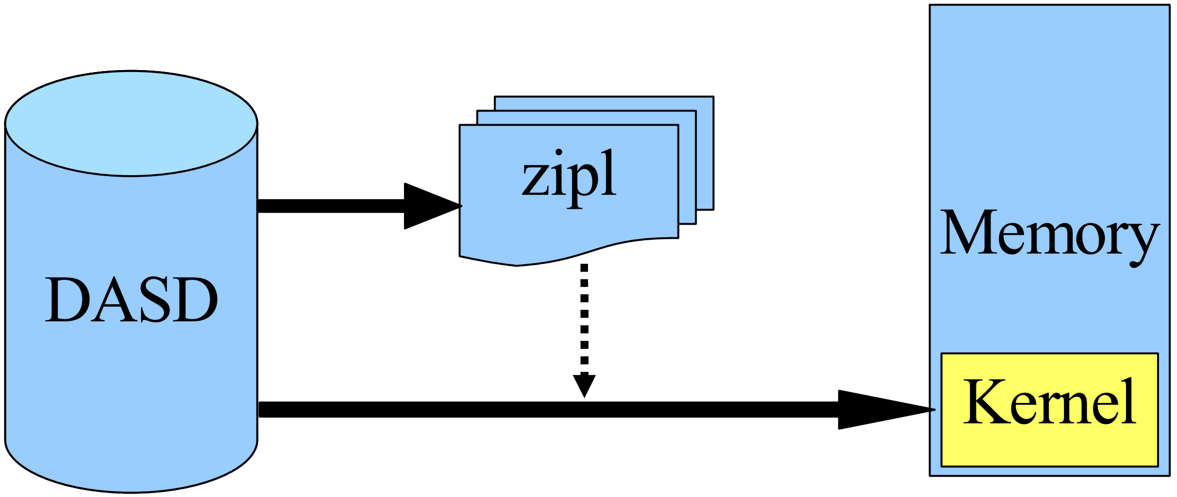
Filesystems

- A logical structure built within a disk partition to manage files
- Many kinds of filesystems are supported
- There is one **root** filesystem: the base of the directory tree
- A filesystem of any type may be **mounted** on a directory
- Mounting is how new storage devices are added
- Unreferenced filesystems may be unmounted



The Boot Loader

- z/VM IPLs a Boot Loader from DASD
- zipl(8) is the boot loader for zSeries Linux
- Knows where to find the kernel within the Linux filesystem
- Passes kernel command-line options
- Configured in `/etc/zipl.conf` [zipl.conf(5)]
- Uses the eckd0 program to store the subchannel address
- Reads kernel file into memory, jumps to entry point



Starting the Kernel

- Kernel is usually in a compressed file
- Beginning of file is program that uncompresses the rest
- Kernel builds its memory pools
- Kernel detects processors, estimates their speed
- Kernel starts its internal threads
- Kernel initializes built-in device drivers
- Drivers do hardware detection
- Drivers can use kernel command line arguments

The Initial RAM disk

- What is an initial RAM disk, and why use one?
  - Extra drivers and setup code
  - Useful when entire kernel won't fit on a floppy (for x86)
  - Lets a distro have a single kernel config across all platforms
  - On zSeries, initrd loads the DASD and zfcp device drivers
- Boot loader told kernel where to find initrd
- Kernel creates a temporary filesystem in memory
- Unpacks the initrd image into that filesystem
- Runs the program `/linuxrc` on it

Initial RAM disks for zSeries

- Loads kernel modules
  - DASD device driver
  - zfcp device driver
  - ext3 filesystem
  - LVM drivers
- Does LVM initialization [see lvm(8), vgscan(8)]
- Mounts the real root filesystem from DASD
- Makes the real root filesystem be the system root
- The mkinitrd(8) tool creates the initrd image

Finishing Kernel Initialization

- Kernel continues when `/linuxrc` on the initrd ends
- Makes the root filesystem read-only, so it can be checked
- Finds `/sbin/init` and runs it

Init: process number one

- Init(8) is the first user-mode process
- It is the root of the process tree
- All other processes are started by init or its descendants
- Reads its configuration file: `/etc/inittab` [see inittab(5)]
- Invokes rc-scripts [see init.d(7)]
- Manages changes between runlevels

Example inittab file (SuSE SLES 8)

```
# The default runlevel is defined here
id:3:initdefault:

# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
#l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# what to do in single-user mode
ls:S:wait:/etc/init.d/rc S

# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
~~:S:respawn:/sbin/sulogin /dev/ttyS0

# on S/390 enable console login in all runlevels
1:012356:respawn:/sbin/mingetty /dev/ttyS0
```

What is an rc-script?

- Runtime configuration scripts live in /etc/init.d
- Each rc-script manages a distinct service or daemon
- These are shell scripts (but they don't have to be)
- Each accepts a single command as an argument:
  - start: starts the service, initializing some resource
  - stop : stops the service, shutting down some resource
  - restart: stops then starts the service
  - status: tells you what state the service is currently in

What Is A Runlevel?

- A feature of the init(8) program
- Controls which processes are allowed to run
- Change to runlevel *N* with command: `init` *`N`*
- Runs master rc-script (`/etc/init.d/rc`) with new runlevel
  - Stops all rc-scripts not in the new runlevel
  - Starts all rc-script that are in the new runlevel
- Runlevels are implemented by directories containing symbolic links to rc-scripts (`/etc/rc?.d`)
  - K*XXname* stops (kills) the service named *name*.
  - S*XXname* starts the service named *name*.

Traditional Set Of Runlevels

- 0: Halt the system
- 1: Single user mode
- 2: Multi-user mode
- 3: Multi-user with networking
- 4: (unused)
- 5: Multi-user with networking and graphical desktops
- 6: Reboot

Boot-time rc-scripts

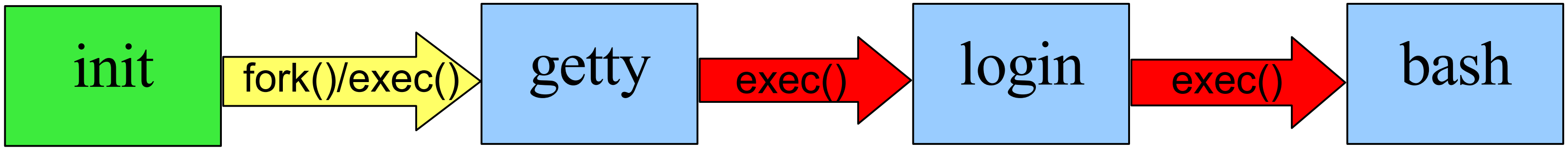
- Run at boot-time from `/etc/init.d/rc` via init(8)
- Bring up user-space (non-kernel) resources:
  - Mount `/proc` and `/sys` pseudo-filesystems (kernel interfaces)
  - Check the root filesystem [fsck(8)]
  - Initialize the LVM subsystem, searching for devices using LVM [vgscan(8)]
  - Check all remaining filesystems [fsck(8)]
  - Enable any swap devices
  - Re-mount root to be writable
  - Mount all other filesystems as described by `/etc/fstab` [fstab(5)]

Service rc-scripts

- Initialize services and daemons for a particular runlevel
- Bookkeeping daemons:
  - cron – periodically run other commands
  - hotplug – detect newly-installed devices (DASD being linked)
  - syslog – collects logging output from other processes
- Network services:
  - interfaces – assign IP addresses or do DHCP, set up routes
  - NFS – mount network filesystems
- Network daemons:
  - sendmail – SMTP daemon listening on port 25
  - xinetd – a meta-daemon listening on many ports, invokes FTP, TELNET...
  - NTP – Network Time Protocol daemon using UDP connections
- Applications:
  - X-Windows – Starts an X display manager to provide user desktops
  - WebSphere – Starts up a web services engine
  - DB2 – Starts one or more database instances

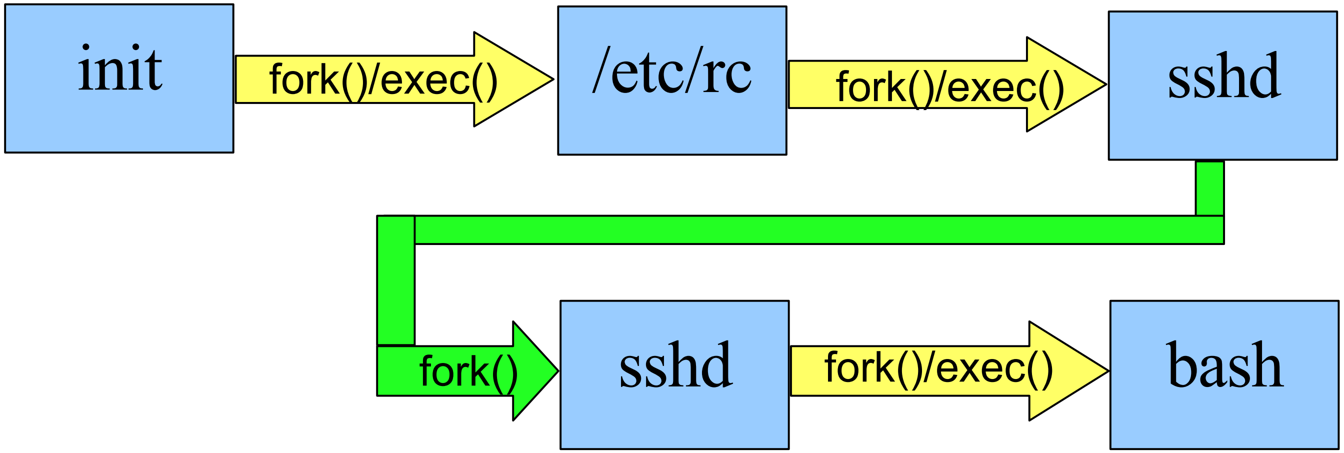
User Logins on Terminals

- Init(8) starts getty(8) processes on attached terminals
- Getty(8) sets up serial tty lines, auto-detecting speed, etc.
- Getty(8) presents a `login:` prompt
- Exec's login(1), giving it the username
- Login(1) gives `password:` prompt, does authentication
- If successful, login(1) invokes the user's shell



User Logins from the Network

- An rc-script starts sshd(8) process
- Sshd(8) listens on port 22 for network connections
- Sshd(8) forks a child process to handle each connection
- SSH client negotiates user credentials with server
- Child sshd authenticates the user credentials
- If successful, child sshd forks the user's shell
- Child sshd continues to encrypt/decrypt data with SSH client



The Linux IPL Procedure

SHARE 2008 San Jose – Session 9274

Contact Information:
Presenter: Ed.MacKenty@RocketSoftware.com
Company: http://www.rocketsoftware.com

Your primary source for enterprise software

Rocket®

MAINSTAR