O14

Bash v. Kor n - What's the Difference?

Pete Davis, Sr. Instructor, IBM

IBM @server xSeries
Technical Conference

Aug. 9 - 13, 2004

Chicago, IL

RETURN TO INDEX

Today's Presentation

- It All Began ...
- The Linux Login Process
- Command Line Command
- Changing the Prompt
- Controlling the Daemons

The Beginnings -- UNIX

- Developed in the late 1960s
- sh
- csh
- ksh
- tcsh
- bash
- zsh

Shell History -- In the Beginning

- In the near beginning there was the Bourne shell ***/bin/sh*** (written by S. R. Bourne). It had (and still does) a very strong powerful syntactical language built into it, with all the features that are commonly considered to produce structured programs; it has particularly strong provisions for controlling input and output and in its expression matching facilities. But no matter how strong its input language is, it had one major drawback; it made nearly no concessions to the interactive user (the only real concession being the use of shell functions and these were only added later) and so there was a gap for something better.

- Along came the people from UCB and the C-shell ***/bin/csh*** was born. Into this shell they put several concepts which were new, (the majority of these being job control and aliasing) and managed to produce a shell that was much better for interactive use. But as well as improving the shell for interactive use they also threw out the baby with the bath water and went for a different input language.

- The theory behind the change was fairly good, the new input language was to resemble C, the language in which UNIX itself was written, but they made a complete mess of implementing it. Out went the good control of input and output and in came the bugs. The new shell was simply too buggy to produce robust shell scripts and so everybody stayed with the Bourne shell for that, but it was considerably better for interactive use so changed to the C shell, this resulted in the situation where people use a different shell for interactive work than for non-interactive, a situation which a large number of people still find themselves in today.

Shell History -- Page Two

- After csh was let loose on an unsuspecting world various people decided that the bugs really should get fixed, and while they where at it they might as well add some extra features. In came command line editing, TENEX-style completion and several other features. Out went most of the bugs, but did the various UNIX operating system manufacturers start shipping ***tcsh*** instead of ***csh***? No, they stuck with the standard C-Shell.

- Eventually David Korn from AT&T had the bright idea to sort out this mess and the Korn shell ***/bin/ksh*** made its appearance. This quite sensibly junked the C shells language and reverted back to the bourne shell language, but it also added in the many features that made the C shell good for interactive work (you could say it was the best of both worlds), on top of this, it also added a some features from other operating. The Korn shell became part of System V but had one major problem; unlike the rest of the UNIX shells it wasn't free, you had to pay AT&T for it.

- It was at about this time that the first attempts to standardize UNIX started in the form of the POSIX standard. POSIX specified more or less the System V Bourne Shell (by this time the BSD and System V versions had become slightly different). Later the standard is upgraded, and somehow the new standard managed to look very much like ***ksh***.

Shell History -- Bourne Again

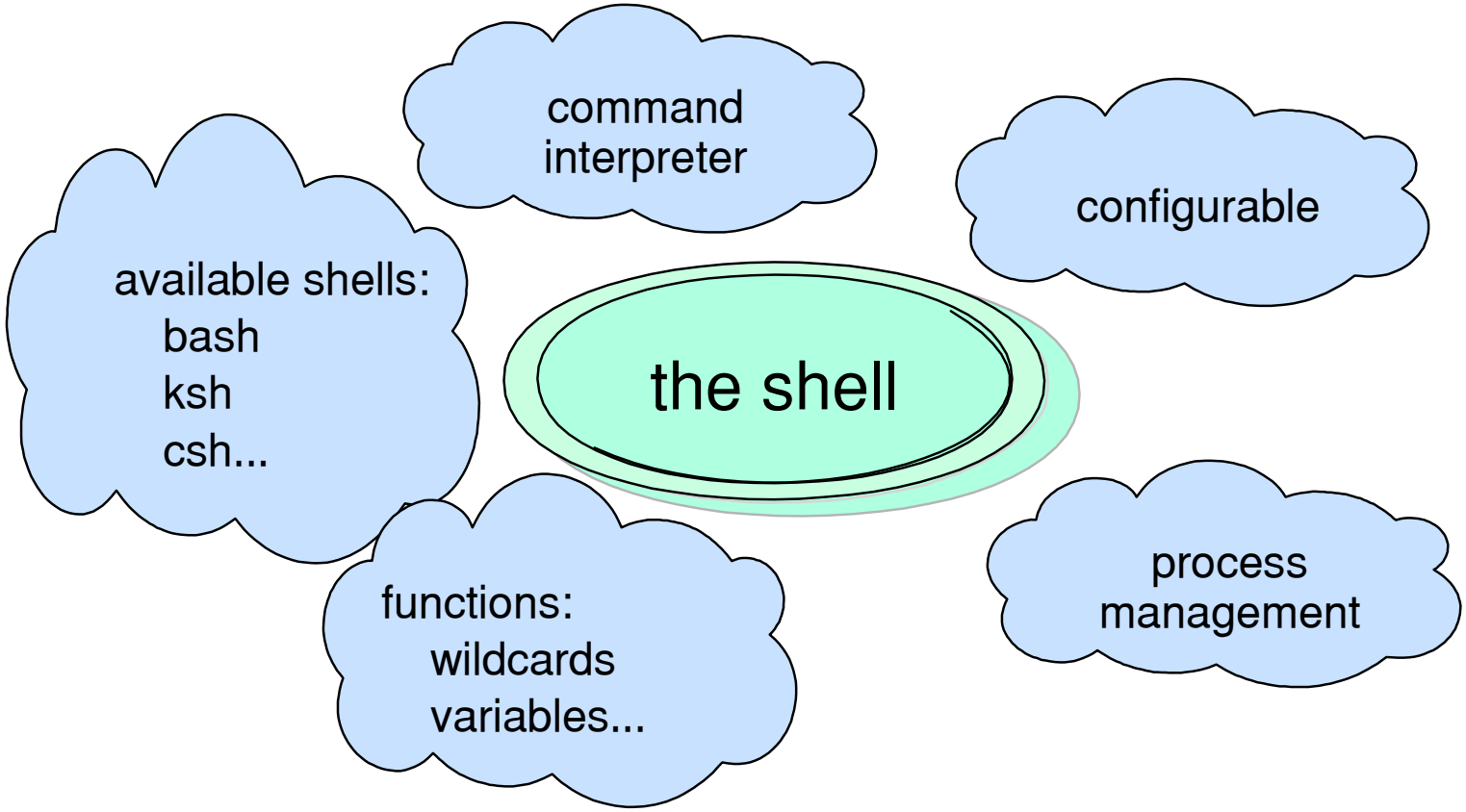
- Also at about this time the GNU project was underway and they decided that they needed a free shell, they also decided that they wanted to make this new shell POSIX compatible, thus ***bash*** (the Bourne again shell) was born. Like the Korn shell, bash was based upon the Bourne shells language. And like the Korn shell, it also pinched features from the C shell and other operating systems. But unlike the Korn shell, ***bash*** is free. Bash was quickly adopted for LINUX (where it can be configured to perform just like the Bourne shell), and is the most popular of the free new generation shells.

- Meanwhile faced with the problem of porting the Bourne shell to Plan 9, Tom Duff revolts and writes ***rc***, he publishes a paper on it, and Byron Rakitzis reimplements it under UNIX. Rc ended up smaller, simpler, more regular and in most peoples opinion, a much better programmed shell.

- The search for the perfect shell still goes on and the latest entry into this arena is ***zsh***. Zsh was written by Paul Falstad while he was a student a Princeton and is a feature packed shell which has so many features that he may not even know all of them.

The Shell

- the user interface to Linux



Shell Features

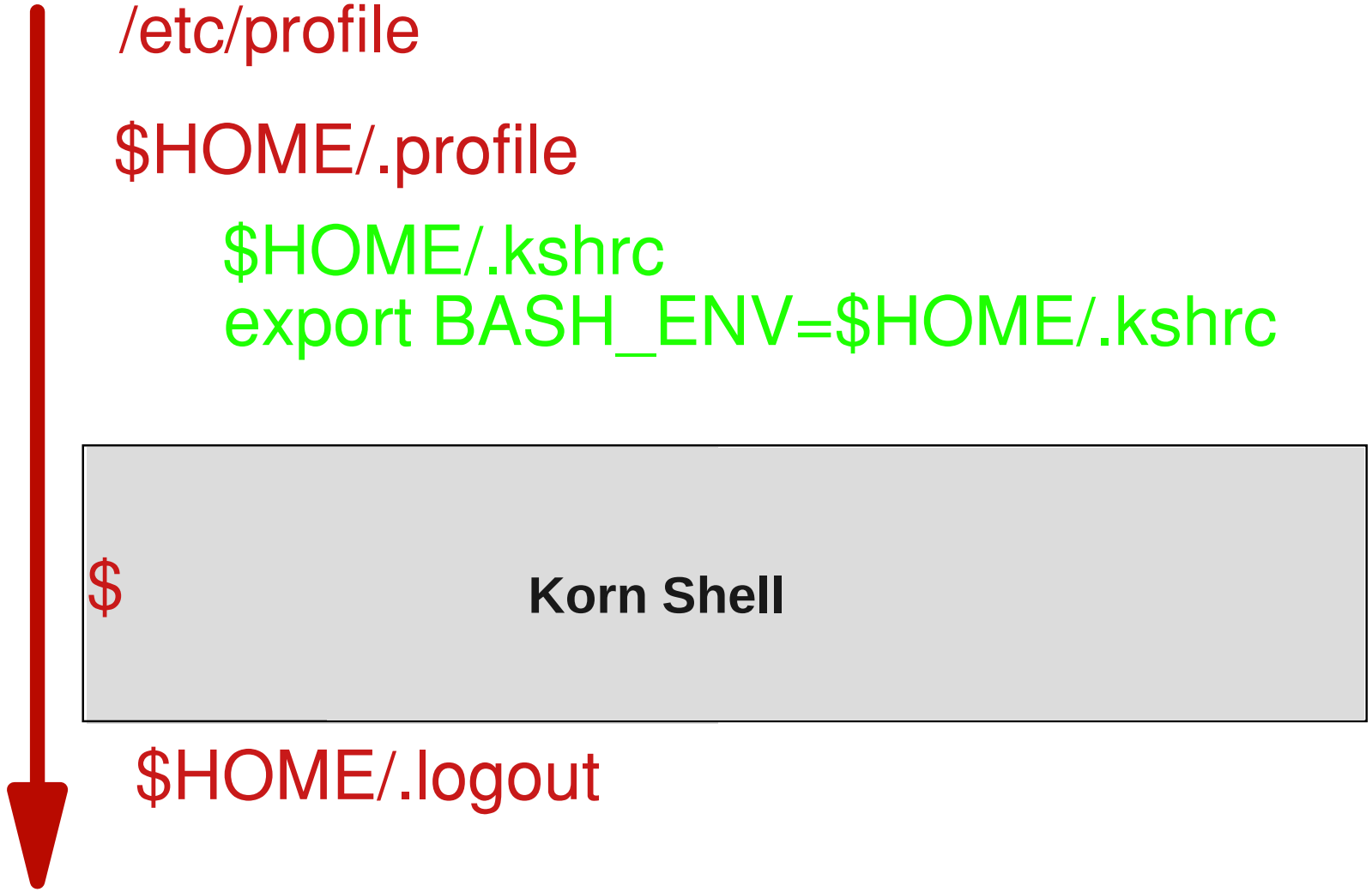
- When the user types a command, various things are done by the shell before the command is actually executed:
  - Wildcard expansion — * ? [ ]
  - Input/Output redirection — < > >> 2>
  - Command grouping — { com1 ; com2; }
  - Line continuation — \
  - Shell variable expansion — $VAR
  - Alias expansion — dir -> ls -l
  - Shell scripting — #!/bin/bash
- For example, the **ls *.doc** command could be expanded to **/bin/ls --color=tty mydoc.doc user.doc** before execution (depending on settings and files present)

ksh v. bash

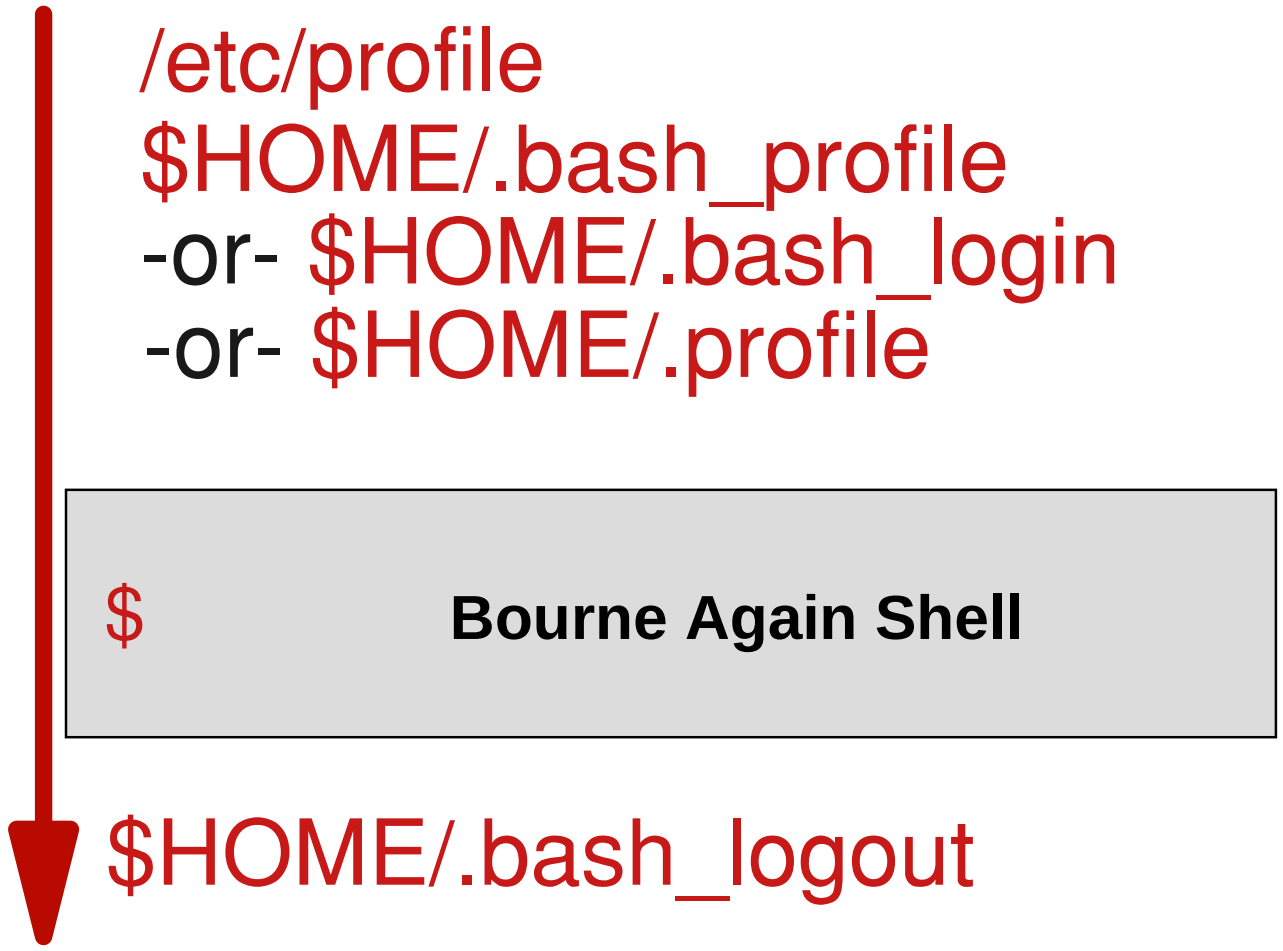
The KornShell is a good alternative to BASH. Its only major drawbacks are that it isn't freely available and is upgraded only every few years.

Learning the bash Shell -- ORA

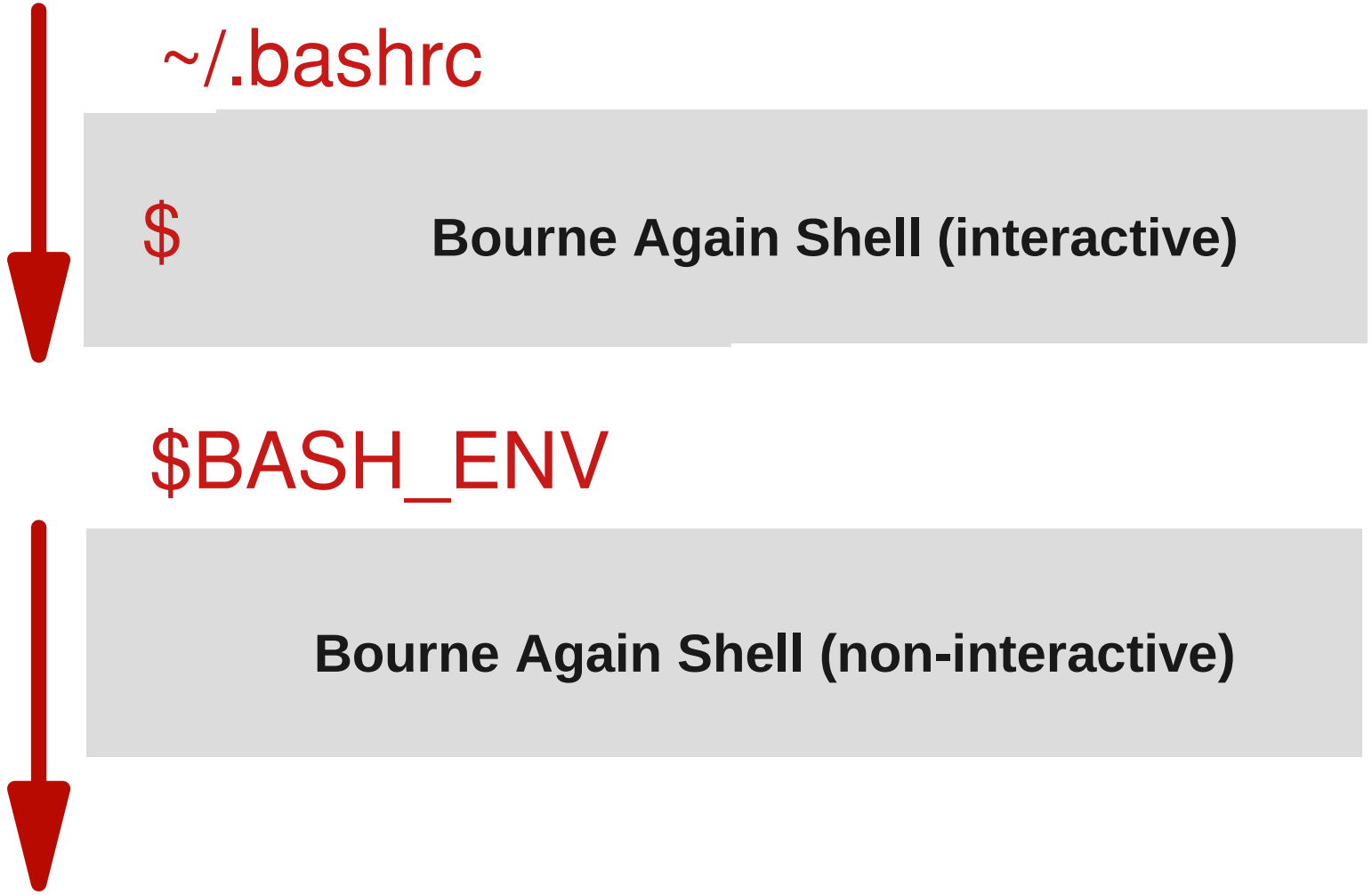
ksh Initialization



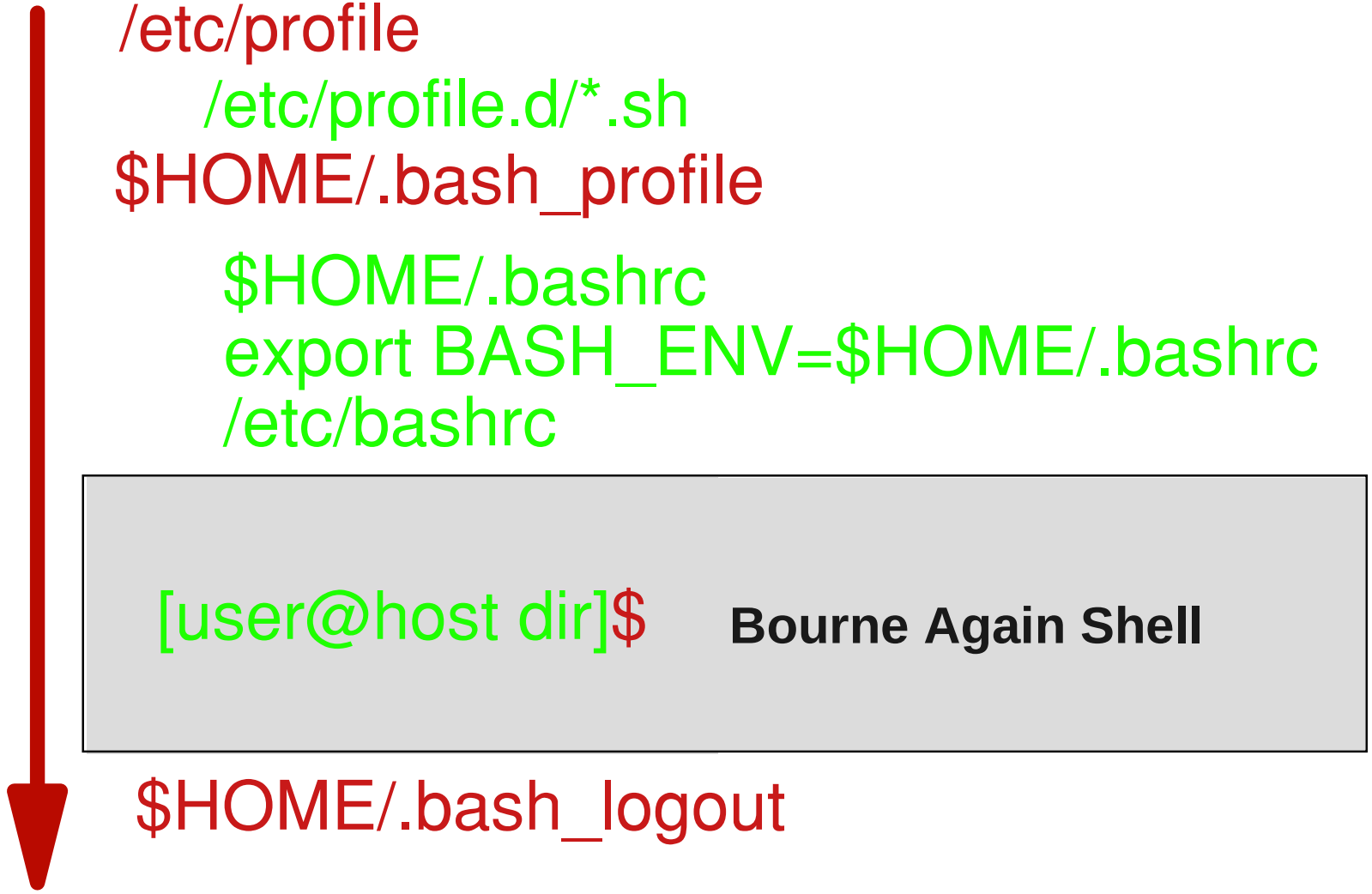
Bash Initialization (Login Shell)



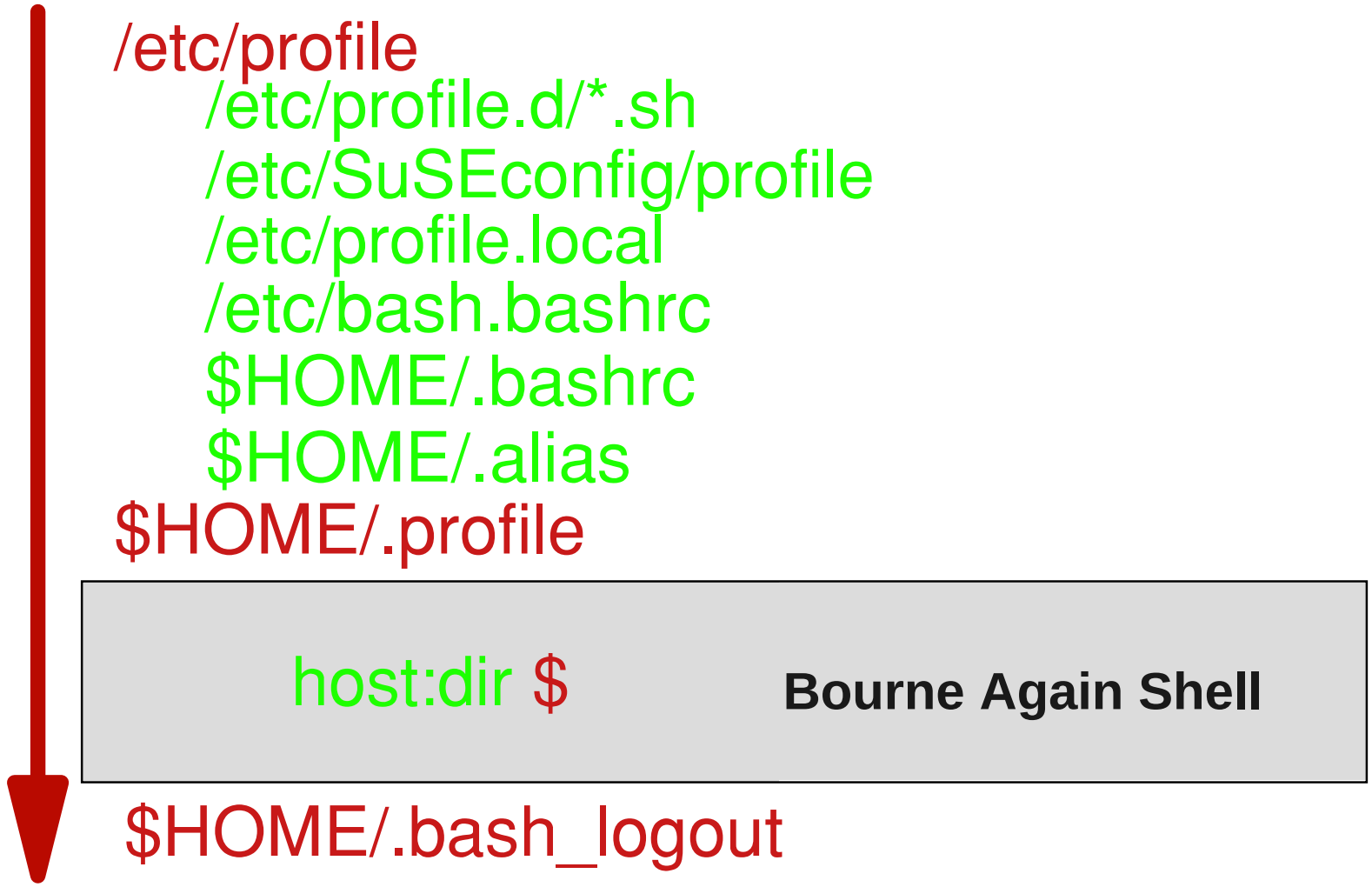
Bash Initialization (non-Login Shell)



Bash Initialization -- RHEL



Bash Initialization -- SuSE



Command Line Command

- set -o vi (ksh)
- h, j, k, l
- CTRL-f
- CTRL-b
- r (fc -e -)
- r old new

- set -o emacs (bash)
- Arrow keys
- Pg-Up
- Pg-Dn
- !n , !string
- ^old^new

Command Line Command

- ESC-ESC
- ESC-\
- ESC-/
- ESC-=
- ESC-*
- ...

- TAB
- ESC-/
- CTRL-R
- $-TAB
- @-TAB
- ~-TAB
- !-TAB
- ...

Command Line Command

- cat file | more
  9999<spacebar>
  b, b, b, b, b, b,
  ....
- tac file | less

Tools & Shortcuts

- for i in 1 2 3...9
- ext filesystem
- jfs
- LVM
- man pages
- smitty

- for i in (seq 1 9)
- ext2 filesystem
- ext3, JFS, ReiserFS
- still maturing
- man, info, HOWTOs
- linuxconf, YAST, LISA, webmin, ...

Tools & Shortcuts

cron - 0-6, 1-12 0,5,10,15,...	Tue, Fri; Mar, Nov; 0/5
tar -z	tar jP
shutdown -r | -h	reboot | halt
swapoff,shutdown -r	swapoff | swapon
unmount	umount

Prompt Mangement

- PS1="$PWD $ "
- ="[\n\u@\h\W]\n\\$"

[pete@home:/data]
$

Bash Prompt Sequences

\D{format}

The format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required

\d the date in "Weekday Month Date" format (e.g., "Tue May 26")
\t the current time in 24-hour HH:MM:SS format
\T the current time in 12-hour HH:MM:SS format
\@ the current time in 12-hour am/pm format
\A the current time in 24-hour HH:MM format

\s the name of the shell, the basename of $0 (following the final slash)
\l the basename of the shell's terminal device name
\V the release of bash, version + patchelvel (e.g., 2.00.0)
\v the version of bash (e.g., 2.00)
\$ if the effective UID is 0, a #, otherwise a $

Bash Prompt Sequences

\u the username of the current user
\h the hostname up to the first `.'
\H the hostname
\w the current working directory
\W the basename of the current working directory
\# the command number of this command
\! the history number of this command
\j the number of jobs currently managed by the shell
\n newline
\r carriage return
\a an ASCII bell character (07)
\e an ASCII escape character (033)
\\ a backslash
\nnn the character corresponding to the octal number nnn
\[\] begin|end a sequence of non-printing characters, which could be used to embed a terminal control sequence

Is It Daemon or Demon ?

- A daemon is a program or process that sits idly in the background until it is invoked to perform its task. It's usually used for applications which passively optimize the o/s environment.
- "daemon" is not an alternative way to spell "demon".

- dai·mon also de·mon or dae·mon
  n. Greek Mythology. An inferior deity, such as a deified hero. An attendant spirit; a genius.
- de·mon, n.
  An evil supernatural being; a devil. A persistently tormenting person, force, or passion: the demon of drug addiction. One who is extremely zealous, skillful, or diligent: worked away like a demon; a real demon at math. Variant of daimon.
- demon
  \De"mon\, n. [F. d['e]mon, L. daemon a spirit, an evil spirit, fr. Gr. ? a divinity; of uncertain origin.] 1. (Gr. Antiq.) A spirit, or immaterial being, holding a middle place between men and deities in pagan mythology.

"Are you a good witch ... ?"

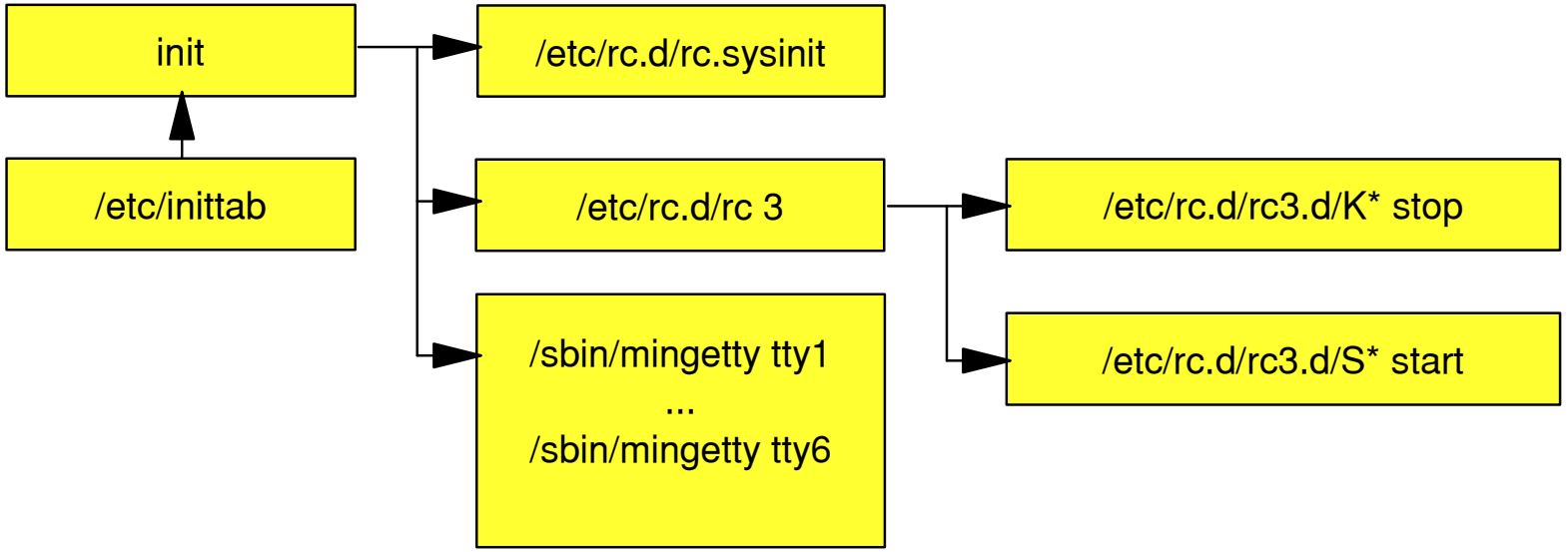
- The demon kind is of an intermediate nature between the divine and the human. --Sydenham.

- Actually, depending on who you talk to, demons and daemons are not the same thing. The dictionary defined "daemon" as the old-english spelling of demon. Spirits who were good called themselves daemons. Demons, try to trick you.

- Daemons are basically what Christians call "guardian angels". Some call them daemons. Going by that respect, demons and daemons are opposites.

- www.thesistersthree.com/forums

Starting Services (System V init style)




```
# ls -l /etc/rc.d/rc3.d
lrwxrwxrwx   1 root root    17 Mar 16 21:17 K10pulse -> ../init.d/pulse
lrwxrwxrwx   1 root root    17 Mar 16 21:17 K10xntpd -> ../init.d/xntpd
.
lrwxrwxrwx   1 root root    17 Mar 16 21:17 S05kudzu -> ../init.d/kudzu
lrwxrwxrwx   1 root root    17 Mar 16 21:17 S10network -> ../init.d/network
lrwxrwxrwx   1 root root    17 Mar 16 21:17 S11portmap -> ../init.d/portmap
.
lrwxrwxrwx   1 root root    17 Mar 16 21:11 S99local -> ../rc.local
```

Controlling the Daemons

- Bash uses xinetd
  - more control, security
  - protocol, port
  - additional log information
  - enable, disable
- example of swat xinetd control file ...

SWAT is the Samba Web Admin Tool.

```
service swat
{
    socket_type         =   stream
    protocol            =   tcp
    user                    =   root
    server              =   /usr/sbin/swat
    only_from           =   127.0.0.1
    disable             =   yes
    port                =   9098
    log_on_success      +=  DURATION USERID
    log_on_failure      +=  USERID
}
```

Controlling the Daemons

- chkconfig <service> list | on | off
- service <service> start | stop | restart ...
- redhat-config-services

-- or --

- rc<service> start | stop | restart ...
- yast2

chkconfig --list

```
server:~ # chkconfig --list | grep -v boot | head -26
SuSEfirewall2_final       0:off   1:off   2:off   3:off   4:off   5:off   6:off
SuSEfirewall2_init        0:off   1:off   2:off   3:off   4:off   5:off   6:off
SuSEfirewall2_setup       0:off   1:off   2:off   3:off   4:off   5:off   6:off
acct                      0:off   1:off   2:off   3:off   4:off   5:off   6:off
acpid                     0:off   1:off   2:off   3:off   4:off   5:off   6:off
alsasound                 0:off   1:off   2:on    3:on    4:off   5:on    6:off
apache                    0:off   1:off   2:off   3:on    4:off   5:on    6:off
apmd                      0:off   1:off   2:off   3:off   4:off   5:off   6:off
atalk                     0:off   1:off   2:off   3:off   4:off   5:off   6:off
atd                       0:off   1:off   2:on    3:on    4:off   5:on    6:off
autofs                    0:off   1:off   2:off   3:on    4:off   5:on    6:off
bgpd                      0:off   1:off   2:off   3:off   4:off   5:off   6:off
bzflagserver              0:off   1:off   2:off   3:off   4:off   5:off   6:off
cron                      0:off   1:off   2:on    3:on    4:off   5:on    6:off
cups                      0:off   1:off   2:off   3:off   4:off   5:off   6:off
dhcpd                     0:off   1:off   2:off   3:on    4:off   5:on    6:off
distccd                   0:off   1:off   2:off   3:off   4:off   5:off   6:off
dvb                       0:off   1:off   2:off   3:off   4:off   5:off   6:off
fbset                     0:off   1:on    2:on    3:on    4:off   5:on    6:off
gpm                       0:off   1:off   2:on    3:on    4:off   5:off   6:off
hotplug                   0:off   1:on    2:on    3:on    4:off   5:on    6:off
hwscan                    0:off   1:off   2:on    3:on    4:off   5:on    6:off
inn                       0:off   1:off   2:off   3:off   4:off   5:off   6:off
ipxmount                  0:off   1:off   2:off   3:off   4:off   5:off   6:off
ipxrip                    0:off   1:off   2:off   3:off   4:off   5:off   6:off
ircd                      0:off   1:off   2:off   3:off   4:off   5:off   6:off
server:~ #
```

chkconfig apache

```
server:~ # chkconfig --list apache
apache                  0:off   1:off   2:off   3:on    4:off   5:on    6:off
server:~ # chkconfig apache off
server:~ # chkconfig --list apache
apache                  0:off   1:off   2:off   3:off   4:off   5:off   6:off
server:~ # chkconfig apache on
server:~ # chkconfig --list apache
apache                  0:off   1:off   2:off   3:on    4:off   5:on    6:off
server:~ #
```

Starting and Stopping Services Manually

- Scripts in init.d directory can be used to start/stop services manually
- On Red Hat, the **service** command calls this script
- On SuSE, the **rc{service}** command calls this script
- Default options:
  - status -- start
  - stop -- restart
- Other options may also be available

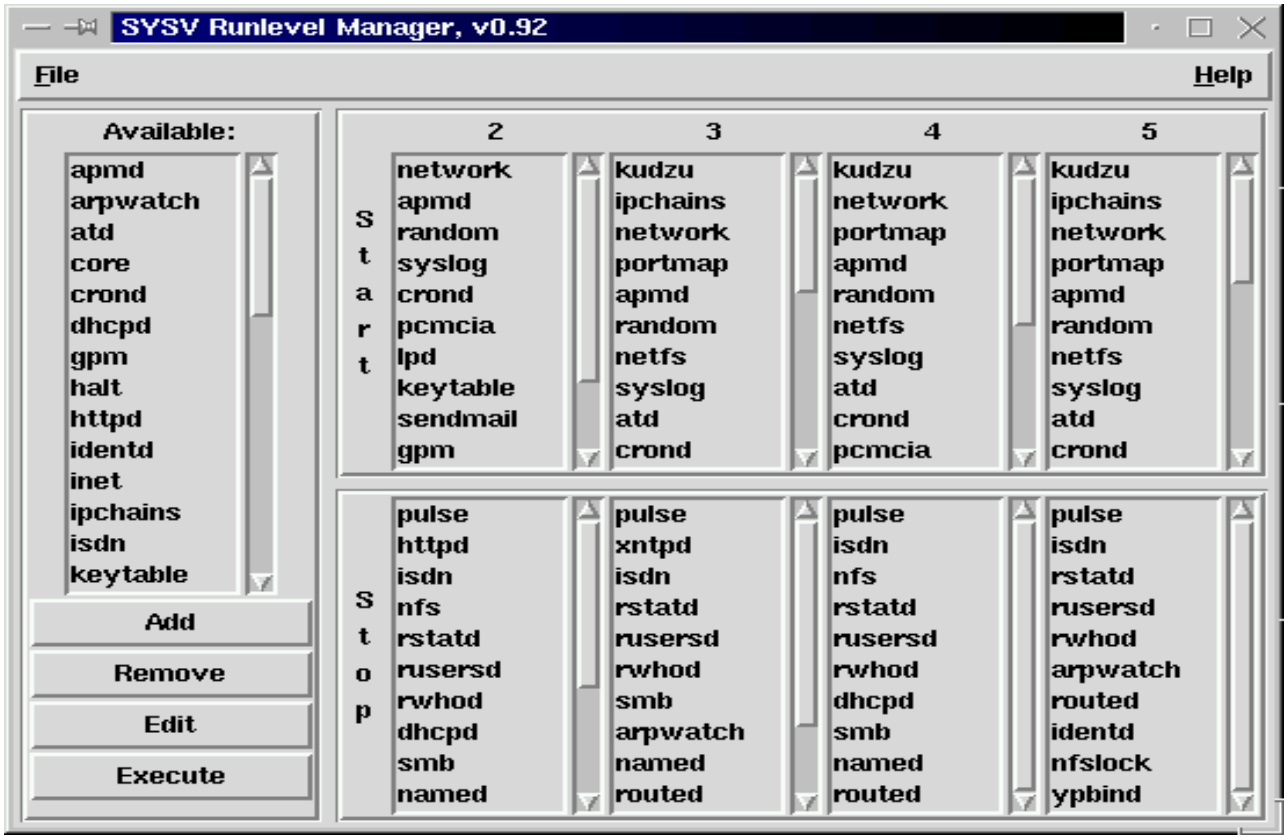
```
# service syslog restart
Shutting down kernel logger:                [ OK ]
Shutting down system logger:                [ OK ]
Starting system logger:                     [ OK ]
Starting kernel logger:                     [ OK ]
```

rcapache

```
server:~ # rcapache
Usage: /usr/sbin/rcapache {start|stop|status|try-restart|restart|force-reload|reload|probe}
server:~ # rcapache status
Checking for httpd:                                                    running
server:~ # rcapache stop
Shutting down httpd                                                    done
server:~ # rcapache status
Checking for httpd:                                                    unused
server:~ # rcapache start
Starting httpd [ Mailman PHP4 ]                                        done
server:~ # rcapache status
Checking for httpd:                                                    running
server:~ # rcapache reload
 Mailman PHP4 Reload httpd                                             done
server:~ # rcapache restart
Shutting down httpd                                                    done
Starting httpd [ Mailman PHP4 ]                                        done
server:~ #
```

Configuring Services per Runlevel

```
# ntsysv
# chkconfig
# ksysv
# serviceconf
# yast
```



To change runlevels use **init <runlevel>** or **telinit <runlevel>**

Summary

- It All Began ...
- The Linux Login Process
- Command Line Command
- Change the Prompt
- Controlling the Daemons

RETURN TO INDEX