# Introduction to Shells[*]

Boris Veytsman

June 28, 2001

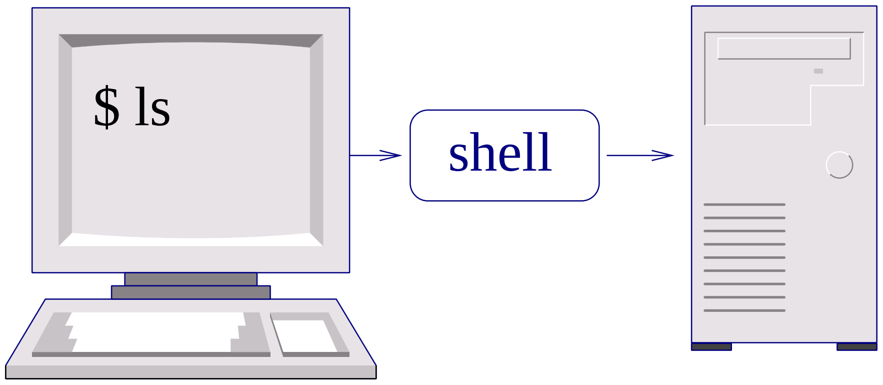
[*]This document contains lecture notes for informal Unix seminar for ITT AES employees (Reston, VA). No information in this document is either endorsed by or attributable to ITT. This document contains no ITT Privileged/Proprietary Information.

ITT Industries
*Engineered for life*

## What Is A Shell?

It is easier to port a shell than a shell script.
*Larry Wall*

Interpreter between you and the computer.



Everything you type is taken care of by shell!

## Two Ways to Use Shell

There are two ways to write error-free programs; only the third one works.

**Interactive:** you write something, and the system executes it immediately. You need:

1. Good process control mechanism
2. Good history and command line editing facilities

**Batch:** you write a script, and the system executes it. You need:

1. Good programming facilities (language!)
2. Good redirection facilities

ITT Industries
*Engineered for life*

◀◀ ◀ ▶ ▶▶

## The Shells You Choose

A UNIX saleslady, Lenore,
Enjoys work, but she likes the beach more.
She found a good way
To combine work and play:
She sells C shells by the seashore.

See http://www.faqs.org/faqs/unix-faq/shell/shell-differences/

**Bourne shell:** */bin/sh*. Very good language, great redirection. Bad for interactive use. 90% scripts use */bin/sh*

**C shell:** */bin/csh*. Tried to mimic C language. Badly. Most implementation buggy. Others are brain dead. Better for interactive use. No good for scripts.

ITT Industries
Engineered for life

**New C shell:** */bin/tcsh*. Bug fixed, new features added. Unfortunately, non standard.

**Korn shell:** */bin/ksh*. Tried to combine Bourne and C shells. Bourne-like syntax, C-like interactive features. Unfortunately, not free and *not* bundled with commercial systems.

**Bourne Again Shell:** */bin/bash*. A free gift from GNU. Compatible with */bin/sh*, but with nice features from C shell, */bin/ksh* and other shells. Good for anything.

**Experimental shells:** */bin/zsh*, */bin/rc* & others.

1. If you are a novice or have time to learn—use *bash*

2. *Never* write scripts in *csh*—see the famous paper by Tom Christiansen, `http://www.perl.com/pub/language/versus/csh.html`.

   > Do yourself a favor, and if you *have* to write a shell script, do it in the Bourne shell.

3. If you do not have *bash*, try *ksh*

4. If you are a *csh* veteran, try *tcsh*. Still, you might benefit form *sh* or *bash*

## Command Line History

We learn from history that we do not learn from history. *Georg Hegel*

Interactive work requires reuse of commands. This is called *command line history*.

## Setting Up

Setting the number of commands to remember

```
HISTSIZE=100 # bash, ksh
set history=100 # csh
```

Remembering the history from session to session:

- Automatic in *bash* & *ksh*
- In *csh* use

```
set savehist=100
```

## Repeating the Last Command

**!!** repeats the last command

```
boris@reston-0491:~$ pwd
/home/boris
boris@reston-0491:~$ !!
pwd
/home/boris
```

You can add something to the command

```
boris@reston-0491:~$ !! > pwd.list
```

ITT Industries
*Engineered for life*

**!*letters*** repeats the last command starting with *letters*

```
boris@reston-0491:~/itt/unix/6_shellsintro$ !pdf
pdflatex 6_shellsintro.tex
```

**!?*letters*?** repeats the last command *containing letters*

```
boris@reston-0491:~/itt/unix/6_shellsintro$ !?shells?
pdflatex 6_shellsintro.tex
```

**history** gives the list of commands with numbers

**!*number*** repeats the command with this *number*

```
boris@reston-0491:~$ history |tail
  497  w
  498  ls
  499  exit
  500  pwd
  501  pwd
  502  history
  503  history |head
  504  history |tail
boris@reston-0491:~$ !500
pwd
/home/boris
```

ITT Industries
Engineered for life

## A real-life example:

```
boris@reston-0491:~$ history |grep gsfc
  118  ssh class.gsfc.nasa.gov
  506  history |grep gsfc
boris@reston-0491:~$ !118
ssh class.gsfc.nasa.gov
```

ITT Industries
*Engineered for life*

## Editing the Last Command

**Switch :p** prints command for editing. The command becomes the last in history!

```
boris@reston-0491:~$ !pdf:p
pdflatex 6_shellsintro.tex
boris@reston-0491:~$
```

**Carets ^. . . ^. . .** substitutes text

```
boris@reston-0491:~$ ls *junk*
junk
boris@reston-0491:~$ ^ls^rm
rm *junk*
boris@reston-0491:~$
```

ITT Industries
*Engineered for life*

**Switch :s** same as carets, but more flexible:

```
boris@reston-0491:~$ ls *junk*
junk
boris@reston-0491:~$ !:s/ls/rm/
rm *junk*
```

It can be sophisticated. . .

```
boris@reston-0491:~$ pdflatex 6_shellsintro.tex
...
boris@reston-0491:~$ !pdf:s/pdflatex/ls -ls/
ls -ls 6_shellsintro.tex
 12 -rw-r--r--  1 boris users 8763 Jun 26 15:15 6_shellsintro.tex
```

## Some Other Odds and Ends

**!$** repeat the last word on the previous line

```
boris@reston-0491:~$ mkdir junk
boris@reston-0491:~$ mv jnk01 !$
mv jnk01 junk
boris@reston-0491:~$ cd !$
cd junk
boris@reston-0491:~/junk$
```

**!:0** repeat the command name

**!*** repeat all arguments

**!-5** repeat last 5 commands

ITT Industries
*Engineered for life*

## Job Control

The difference between a career and a job is about 20 hours a week.

## Background and Foreground

A command you started normally is attached to the tty (*foreground*):

```
boris@reston-0491:~$ xeyes
```

Ampersand puts it into *background*:

```
boris@reston-0491:~$ xeyes &
[1] 4076
```

We started *job #1, process #4076*

ITT Industries
Engineered for life

A foreground job can be *killed*

```
boris@reston-0491:~$ xeyes
Ctrl-c
boris@reston-0491:~$
```

It *stopped* and put into background

```
boris@reston-0491:~$ xeyes
Ctrl-z
[2]+  Stopped                 xeyes
boris@reston-0491:~$ bg
[2]+ xeyes &
boris@reston-0491:~$
```

ITT Industries
Engineered for life

## Getting List of Jobs and Processes

```
boris@reston-0491:~$ jobs
[1]   Running                 xeyes &
[2]-  Running                 xbiff &
[3]+  Running                 oclock &
boris@reston-0491:~$ ps
  PID TTY          TIME CMD
 4092 pts/0    00:00:00 bash
 4093 pts/0    00:00:00 xeyes
 4094 pts/0    00:00:00 xbiff
 4095 pts/0    00:00:00 oclock
 4096 pts/0    00:00:00 ps
boris@reston-0491:~$
```

We can obtain list of *all* processes:

- SysV: *ps -ef*
- BSD: *ps -aux*

## Killing Jobs and Processes

```
boris@reston-0491:~$ jobs
[1]   Running                 xeyes &
[2]-  Running                 xbiff &
boris@reston-0491:~$ ps
  PID TTY          TIME CMD
 4092 pts/0    00:00:00 bash
 4093 pts/0    00:00:00 xeyes
 4094 pts/0    00:00:00 xbiff
 4096 pts/0    00:00:00 ps
boris@reston-0491:~$ kill %1
[1]   Terminated              xeyes
boris@reston-0491:~$ kill 4094
[2]-  Terminated              xbiff
boris@reston-0491:~$
```

## A Real Life Example: Run Away Netscape

```
boris@reston-0491:~$ ps aux |grep netscape
boris     9607  0.0 10.6 34488 26988 ?      \
 S    Jun25   0:57 /usr/lib/netscape/477/communicator\
/communicator-smotif.real http://www.de.ittind.com/
boris@reston-0491:~$ kill 9607
boris@reston-0491:~$
```

## Ways to Kill. . .

To send a signal to a program:

```
kill -HUP %1
kill -1   %1
```

Some common signals

| Signal | Value | Meaning |
|---|---|---|
| HUP | 1 | Hangup. Often causes reloading |
| KILL | 9 | Kills politely |
| TERM | 15 | Kills impolitely |
| STOP | 19 | Suspends |
| CONT | 18 | Allows to continue |

## Input, Output and *nohup*

```
nohup rm -fr / &
```

Command, started form command line, use your keyboard as input and your screen as output.

What if you put the command into background?

1. If it needs input, it stops

2. Output is still your screen

ITT Industries
*Engineered for life*

```
boris@reston-0491:~$ telnet kenny
Trying 151.190.55.231...
Connected to kenny.reston.aes.de.ittind.com.
Escape character is '^]'.

SunOS 5.6 (kenny)

login:
```

## Does not work in background:

```
boris@reston-0491:~$ telnet kenny &
[1] 4414
boris@reston-0491:~$ Trying 151.190.55.231...
Connected to kenny.reston.aes.de.ittind.com.
Escape character is '^]'.


[1]+  Stopped                 telnet kenny
boris@reston-0491:~$
```

ITT Industries
*Engineered for life*

Sometimes we can wake it up:

```
boris@reston-0491:~$ fg
telnet kenny

SunOS 5.6 (kenny)

login:
```

What happens to our background jobs if we log out?

1. The program receives the *HUP* signal (from hangup. Remember old modems?)

2. If it tries to write to a screen, it dies

What if we want to have it overnight?

```
nohup myprogram &
```

1. The *HUP* signal is not sent

2. Output goes to the file *nohup.out*

ITT Industries
Engineered for life