UNIX Apologetics for the MPE Guru

David Totsch

Account Support Consultant

Hewlett-Packard Company

HP WORLD 2003

Solutions and Technology Conference & Expo

UNIX Apologetics for the MPE Guru

A Historical Perspective

UNIX LIneage

Ken Thompson	MULTICS (MULTiplexed Information & Computing System) Developed a plan for a new file system
Brian Kernighan	UNICS (UNiplexed Information & Computing System) The system only allowed two simultaneous users. Name was eventually transposed to UNIX.
Dennis Ritchie	C Programming Language The first operating system written in a high-level computer language.

Distributed for the cost of media to Universities researching for AT&T.
AT&T prohibited by federal law from selling computers and software.

University of California at Berkeley - Berkeley Software Distribution (BSD)

A Historical Perspective

Consistent Inconsistency

- BSD introduced networking, sockets and some other utilities that were improvements on AT&T's version.
- AT&T continued to develop UNIX for internal use leading to System V
- HP-UX is a typical hybrid system - it has an AT&T System V kernel that incorporates BSD extensions.

A Historical Perspective

Defies Description

What does all of that mean to us?

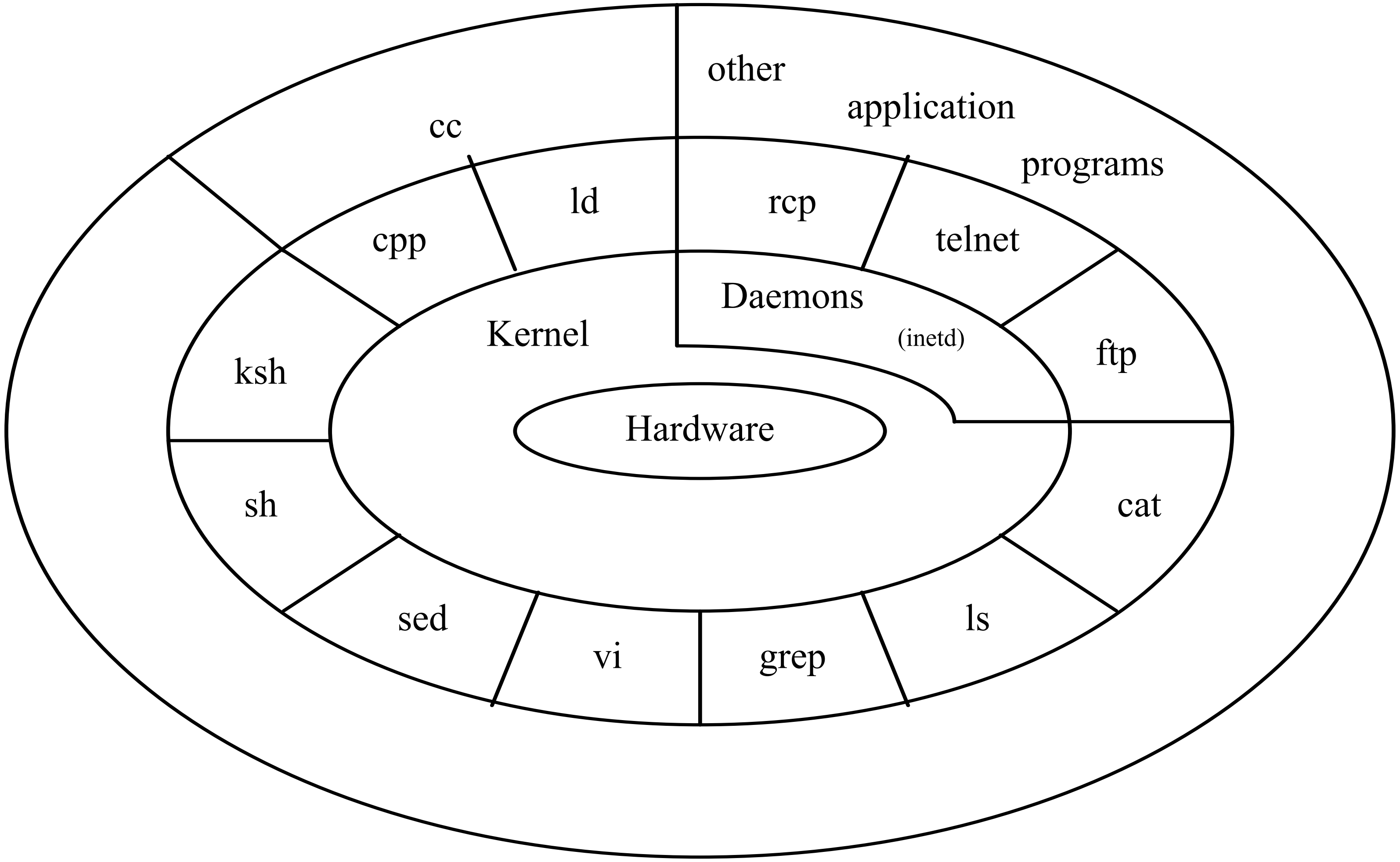
UNIX was developed by programmers for programmers.
UNIX was not intended to be a business system OS.

Therefore

UNIX is “expert friendly”
UNIX is not necessarily “user friendly”
UNIX is user indifferent

A Historical Perspective

Built Upon Itself



A Historical Perspective

Admonishment

Bilingual People don’t memorize languages:

they think in that language.

Don’t be Preoccupied with what other environments do:

Think UNIX!

Getting On The System

What Happens When You Log In

Getty	login:
login	password:
shell	
cd	**HOME**
/etc/profile	**<system settings>**
${HOME}/.profile	**<personal customizations>**
	$
	$ exit
getty	login:

The Shell Environment

- Named Parameters (variables)
  - Where to find stuff
  - Hints to programs
  - Prompt strings
  - Status Information
  - Command Recall
- Functions
- Aliases

The Shell Environment

Where To Find Stuff

- PATH
  - Where to find commands, set in /etc/profile, you append
  - Append with PATH=$PATH:/newpath:/other/new/path
  - System stuff first, then your special directories
- MANPATH
  - Where to find manual pages, set in /etc/profile, you append
- CDPATH
  - Where to find directories, you will set in $HOME/.profile
  - If directory is not in CWD, CDPATH is searched
  - Colon separated list (like PATH)

The Shell Environment

Hints To Programs

- TERM
  - Set in /etc/profile with eval `ttytype -s -a`
  - Used by full-screen commands (e.g. vi)
  - Other terminal settings: stty erase ^H
- LINES, COLUMNS
  - Used by vi, pg, more, etc.
- TZ
  - Timezone set in /etc/profile, used by date, ls, etc.
- LPDEST
  - You will set this one in $HOME/.profile

The Shell Environment

Prompt Strings

- PS1
  - Command line prompt string
  - Be creative
  - PS1=`hostname`':!:$PWD> '
- PS2
  - When you don't complete a command…
- PS3
  - Used as the prompt for shell intrinsic select command
- PS4
  - Used during shell tracing

The Shell Environment

Status Information

- PWD
  - The present working directory (output of pwd cmd)
- ?
  - Integer exit status of last command
- _
  - The last argument of the last command

The Shell Environment

Command Recall

- HISTFILE
  - Points to file that holds the command history
  - HISTFILE=$HOME/.sh_history is BORING!
  - Try using the output of tty, or date
- HISTSIZE
  - How many commands to keep in HISTFILE
- EDITOR, VISUAL
  - What editor you want to use, usually vi.

The Shell Environment

Checking What You Have

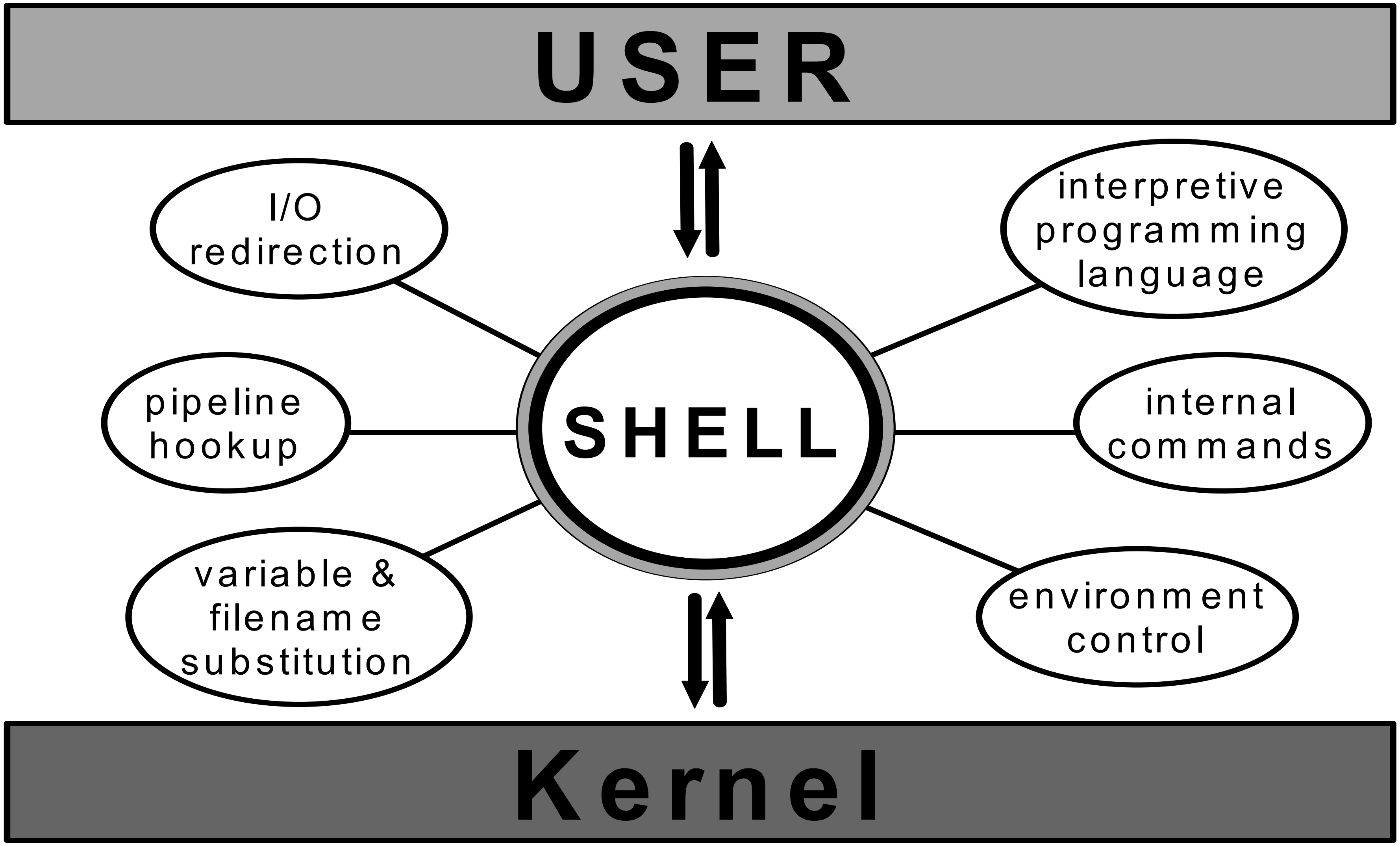
- echo $var
- set
- env

The Shell Environment

Functions and Aliases

- Remember, /etc/profile and $HOME/.profile are for the Bourne shell…what do we do with functionality (**syntax**) it does not support…
  - ENV (please do not ever export!)
    - You might want to put functions here
    - Alias rm "rm -I"

The Shell Command Interface



UNIX Command Philosophy

- According to Kernighan and Pike:
  - output of any command should be usable as input for other commands.
  - If no arguments are given, a program should read standard input and write standard output.
  - All information needed by a program should either be contained in the data stream passed to it or specified on the command line.

UNIX Philosophy

Any complex problem can be broken down into a finite number of discrete tasks.

Typing Commands Into The Shell

- UNIX is case sensitive
- Use the “Backspace” key (or Ctrl-h) to erase *
- Use Ctrl-c to erase an entire line.
- Ctrl-c also interrupts (stops) an executing command.

Command-line Syntax

```
command [options] [arguments]
```

- Command
  - Intrinsic (the shell itself knows)
  - Extrinsic (first matching file in $PATH)
  - Single task and does it very well
- Options
  - Single-letters that modify command behavior
- Arguments
  - Additional data that modifies command behavior
  - Usually required by an option, or a simple file name

Poking Around
Where Am I?

- pwd(1)
  - Or you can make $PWD part of your prompt…
- ls(1)
  - llong listing
  - t   sort by time
  - rt   reverse sort by time (latest last)
  - p   follow directories with /
  - F   / dir, * executable, @ symlink, | FIFO
- cd

Files and Directories

- File
  - String of characters (maybe line-feeds)
  - You impose format
  - All-defining "inode"
- Directory
  - A file! (well, a system formatted one…)
  - Inode,filename
  - Dot
  - Dot dot
  - Hidden Files
- Others (limited set defined by UNIX)

Manipulating Files

Creation

- mkdir(1)
- touch(1)
- Shell redirection (>, >>)
- prealloc(1)
- Programatically (e.g. vi)

Manipulating Files

Examine Contents

- cat(1)
- Before you cat!
  - ls -l
  - file(1)
  - what(1)
- pg(1), more(1), and view(1)
- head(1) and tail(1)
- wc(1)

Manipulating Files

Cleaning Up After Yourself

- rmdir(1)
- rm(1)

Manipulating Files

Moving Things Around

- mv(1)
- cp(1)
- ln(1)
- cpio(1), tar(1), and fbackup(1)

Manipulating Files

Permissions

- Quit Groaning, you knew I had to go here…
- Three Sets
  - User (owner)
  - Group
  - Other (everybody else)
- Three Permissions
  - Read
  - Write
  - Execute/Traverse

Manipulating Files

Permissions (continued)

- Defaults
  - Umask
- chown(1)
  - `chown davidt:sysadmin thatfile`
- chgrp(1)
  - `chgrp sysadmin thatfile`
- chmod(1)
  - `chmod 755 thatdir` ← absolute
  - `chmod g+w thatfile` ← relative

Manipulating Files

Permissions (continued)

NOTICE!

You do not have a complete understanding of the permissions on a file until you trace the directory path to where no-one has access to the parent directory.

Manipulating Files

More Than One

- * matches any sequence of characters (including no characters at all).
- [ ] character class. [A-Z] [0-9] [abc] Matches any char in the list to a single char position.
- ? matches any single character.

These do NOT match “hidden files”.

Manipulating Files

File Name Generation Examples

```
a1      b1      c1      eg.awk          eg.sh           first.doc
a2      b2      c2      eg.data         first.awk       first.sh
a3      b3      c3      eg.doc          first.data
```

```
$ ls *.doc
eg.doc     first.doc
$ ls eg.*
eg.awk   eg.data  eg.doc   eg.sh
$ ls ?????
eg.sh
$ ls ?[0-9]
a1  a2  a3  b1  b2  b3  c1  c2  c3
$ ls [!abc]*
eg.awk      eg.doc      first.awk   first.doc
eg.data     eg.sh       first.data  first.sh
```

Manipulating Files

Extended File Name Generation

?(ptrn_list)	optionally match one pattern
*(ptrn_list)	zero or more
+(ptrn_list)	one or more
@(ptrn_list)	exactly one
!(ptrn_list)	exception
ptrn_list	file name generation equations separated by “|”

Manipulating Files

Extended File Name Generation Examples

```
eg.awk        eg.sh         first.doc
eg.data       first.awk     first.sh
eg.doc        first.data    second.test
```

```
$ ls *.@(sh|awk)
eg.awk        eg.sh         first.awk     first.sh

$ ls *.!(sh|awk)
eg.data       eg.doc        first.data    first.doc

$ls *.@(??|???)
eg.awk        eg.doc        eg.sh         first.awk
first.doc     first.sh
```

The Great vi(1)

- You are most comfortable with what you grew up with
- Your mother didn't encourage you to try different editors like she did vegetables
- Can I take a moment to help you with vi(1)?
  - Moded editor (input/command)
  - hjkl (stop thinking, just do)
  - I know you were an expert on another editor, but you didn't learn that one in one or two sittings
  - REMEMBER: vi(1) is a front-end to the line editor ex(1)
  - Set and .exrc are your friends

Writing Programs

- c, c++
- shell
- awk
- Perl
- The Program "Hello World"

Program Execution

- The first two bytes
  - file(1)
  - #!
  - File names are irrelevant
- C runs directly
- Shell is interpreted
- Awk is interpreted
- Perl is interpreted

Facility Abilities

- Development time / maintainability
- Execution Speed
- Portability
- Variables (data types)
- Math (integer/floating point)
- Regular Expressions
- System Calls
- Manipulating Records (fixed-length, variable-length, delimited)

Some More About Processes

- What am I doing?
  - ps(1)
- What are my neighbors doing?
  - ps -ef
  - ps -fu <user>
- Who are my neighbors?
  - who(1), w(1)
- How busy is the system?
  - uptime(1)

Being Kind to the Processor

- Backgrounding Jobs
  - &
  - jobs
  - fg
- Scheduling Jobs
  - at(1)
  - cron
  - batch(1)
  - nice(1)
  - nohup(1)

Commands You Will Love to Hate

- grep(1)
- cut(1)
- paste(1)
- sort(1)
- join(1)
- wc(1)
- sed(1)
- find(1)
- awk(1)

grep(1)

grep [*options*] *RE* [*file*]

```
$ grep 'lj..0245' *.cfg
nbsgate5.cfg:MEMBER=lj410245
nbsgate7.cfg:MEMBER=lj420245
$ grep -c lj..0245 *.cfg
2
```

Queries data stream for lines matching *RE*

- -v invert output (print lines not matching *RE*
- -i ignore case
- -n report line numbers
- -c count lines with RE
- -q work quietly (no output)

RE Regular Expression

cut(1)

cut [*options*] [*file*]

```
$ grep 'lj..0245' \
   nbsgate?.cfg |  \
   cut -d= -f2
lj410245
lj420245
```

options:

-c*list*	cut by characters
-f*list*	cut by fields
-d	field delimiter
list	-3,7,9-12,14-

paste(1)

paste [*file1*] [*file2*]

```
$ cat fnames
Zoe
Lori
Mark
Dawn
Jon
Erin
```

```
$ cat lnames
Smith
Adams
Brown
Dey
Smith
Smith
```

```
$ paste -d':' fnames lnames
Zoe:Smith
Lori:Adams
Mark:Brown
Dawn:Dey
Jon:Smith
Erin:Smith
```

paste *file1* and *file2* together vertically by line

paste - -

sort(1)

sort [*options*] [*file*]

```
$ cat data
Jon
Zoe
mille
Jon
Brad
Gwen
$ sort -u data
Brad
Gwen
Jon
Zoe
mille
```

options:

-f	fold letters uppercase
-n	numeric sort
-r	reverse order
-t	field separator
-u	unique
-k	restricted key

sort(1) <example> (input data)

data2

```
Zoe:Smith:268:Technology
Lori:Adams:178:Finance
Mark:Brown:22:Marketing
Dawn:Dey:184:Accounting
Jon:Smith:1:Accounting
Erin:Smith:483:Technology
```

sort(1) <example> (lastname, firstname in reverse)

```
$ sort -t: -k 2,2 -k 1,1r data2
Lori:Adams:178:Finance
Mark:Brown:22:Marketing
Dawn:Dey:184:Accounting
Zoe:Smith:268:Technology
Jon:Smith:1:Accounting
Erin:Smith:483:Technology
```

sort(1) <example> (sort the number field)

```
$ sort -t: -k 3,3 data2
Jon:Smith:1:Accounting
Lori:Adams:178:Finance
Dawn:Dey:184:Accounting
Mark:Brown:22:Marketing
Zoe:Smith:268:Technology
Erin:Smith:483:Technology
```

Oops!

sort(1) <example> (numeric sort the number field)

```
$ sort -t: -n -k 3,3 data2
Jon:Smith:1:Accounting
Mark:Brown:22:Marketing
Lori:Adams:178:Finance
Dawn:Dey:184:Accounting
Zoe:Smith:268:Technology
Erin:Smith:483:Technology
```

Ah!

join(1)

join [*options*] [*file1*] [*file2*]

```
# Example on next page
```

options:

-j1 *m*	join on field m of *file1*
-j2 *n*	join on field *n* of *file2*
-t *c*	field separator is *c*
-o *list*	list identifies fields from file to include in the output

file1 and *file2* must be sorted on the joined fields

join(1) (example)

```
$ cat d5
Dawn:Dey:184:A
Jon:Smith:1:A
Lori:Adams:178:F
Mark:Brown:22:M
Erin:Smith:483:T
Zoe:Smith:268:T
```

```
$ cat d4
A:Accounting
F:Finance
M:Marketing
T:Technology
```

```
$join -t: -j2 1 -j1 4 -o 1.1, 1.2, 1.3, 2.2 d5 d4
Dawn:Dey:184:Accounting
Jon:Smith:1:Accounting
Lori:Adams:178:Finance
Mark:Brown:22:Marketing
Erin:Smith:483:Technology
Zoe:Smith:268:Technology
```

wc(1)

wc -l ${FILE}

```
COUNT=$(wc -l ${DATA})
echo $COUNT
75 MyFile
COUNT=$(wc -l <${DATA})
echo $COUNT
75
```

counts lines, words, characters in a file

-l	count lines
-w	count words
-c	count characters

sed(1)

sed [-*n*] script file

```
$ sed -n 500,505p data
$ sed -n '500,$p' data
```

options:

-n only print when told

-e to specify +1 scripts

-f to specify file that contains scripts to use

You will need to quote the script when it contains white space, or $ or…

find(1)

find *start_path options*

```
# List files in
# ${LOGDIR} over 30
# days old
find ${LOGDIR}     \
  -mtime +30       \
  -type f          \
  -print
```

traverses a directory structure printing file names

start_path is the directory to begin looking from

options are many and can be complicate, read the man(1) page.

awk(1)

awk [-Ffs] [-v var=value] [program | -f progfile ...] [file ...]

```
awk -F: '/dlt/{print $3}'
   /etc/passwd
```

options:

-F specify field separator, which can be FNG

-v set variable

-f specify file that contains program

Also has functions that assist process fixed-width records

Putting It All Together

- I/O Redirection
  - >
  - >>
  - 2>
  - 2>&1
- Pipelines
  - Cmd | cmd1 | cmd2
  - tee(1)

Putting It All Together
An Example

Get PID for Oracle listener

```
ps -ef | grep listener | \
grep -v grep | awk '{print $2}'
```

or

```
ps -ef | grep [l]istener | awk '{print $2}'
```

even better

```
ps -ef | awk '/[l]istener/ {print $2}'
```

Still better

```
ps -fu oracle | awk '/[l]istener/ {print $2}'
```

Where Do I Put Stuff
Where Do I Find Stuff

- /
- /opt
- /var/opt
- /var/tmp
- /tmp
- /etc
- /usr
- /sbin

Regular Expresssions

- Take an aspirin, this can become painful…
- Why should I care about REs?
  - It is a major strength of UNIX
  - Formulate powerful queries
  - Automate sophisticated text edits
  - A valuable tool to include in your UNIX skill set
  - Express yourself creatively

Basic
Regular Expressions

[abc]	collating sequence (an "a", a "b" or a "c")
[^abc]	non-collating sequence (anything but, an "a", "b", or "c")
[a-z]	collating sequence specified as range
[: :]	character class (eg. [:upper:])
*	zero or more
.	any single character
( )	grouping (subexpression)
^ $	anchoring (^ = beginning of line, $ = end of line)
\n	n’th subexpression n = 0-9
\{m,n\}	repetition m = minimum, n=max m/n 0-255

Extended Regular Expressions

+	one or more occurrences of preceding RE
?	zero or one occurrences of preceding RE
|	alternation

An observation:

*	Abbreviation for \{0,\}
+	Abbreviation for \{1,\}
?	Abbreviation for \{0,1\}

!!!!!

```
^(\+|-)?[0-9]*\.?[0-9]+$
```

Why Should I Care About REs?

```
$ cat sample
printf("ENTER YOUR NAME: ");
 printf ( "OPENING FILE: %s\n", file );
printf ( "CAN'T OPEN FILE: %s\n", "/usr/data" );
fprintf(stderr,"CAN'T OPEN %s\n",errfile);
/* comment */ printf("CAN'T OPEN %s\n", errfile);

$ cat sed.script
s/\(^.*[^f]\)printf.*"\(.*\)",\(.*\)/\1fprintf(stderr,"\2",\3/
s/^printf.*"\(.*\)",\(.*\)/fprintf(stderr,"\1",\2/

$ sed -f sed.script sample
printf("ENTER YOUR NAME: ");
 fprintf(stderr,"OPENING FILE: %s\n", file );
fprintf(stderr,"CAN'T OPEN FILE: %s\n", "/usr/data" );
fprintf(stderr,"CAN'T OPEN %s\n",errfile);
/* comment */ fprintf(stderr,"CAN'T OPEN %s\n", errfile);
```

Getting Help

- HPWorld, Peers, Books, HPADMIN, ITRC, WEB
- docs.hp.com
- man(1) ("See Also")
  - man <cmd> | col -b > cmd.man
  - Manual Sections
    - 1 User Commands
    - 2 System Calls
    - 3 Functions and Function Libraries
    - 4 File Formats
    - 5 Miscellaneous Topics
    - 6 Device (Special) Files
    - 1M System Management Commands
    - 9 Glossary

How do I get out'a here?

exit

Where Do I Go From Here?

- Practice, Practice, Practice.
- If you cannot think of at least three ways to do something in UNIX, you just aren’t thinking hard enough.
- Once you are comfortable with the vi editor, go for using command history
- Shell script writing is basically typing the commands you would normally enter into the shell into a file...
- The more commands you know, the closer you are to becoming a “Power User”.

HP WORLD 2003

Solutions and Technology Conference & Expo

Interex, Encompass and HP bring you a powerful new HP World.

interex shared knowledge • shared power

encompass AN HP USER GROUP

hp invent