Open Source on OpenVMS?

It's Easier Than You Think!

Brad McCusker
OpenVMS Engineering
Aug 18, 2004

Introduction

Introduction

- Presenter
  - Brad McCusker
    - OpenVMS C RTL Project Leader
- Topic
  - Porting OpenSource software to OpenVMS
    - GNV
    - CRTL

Unix Portability

- OpenSource Software
- Ease of installation on variety of Unix systems
  - Unix®
- Unix software on OpenVMS
  - C language
  - DEC C Runtime Library (CRTL)
  - GNV (GNU's not VMS)
  - JAVA
- Make OpenVMS just like another Unix

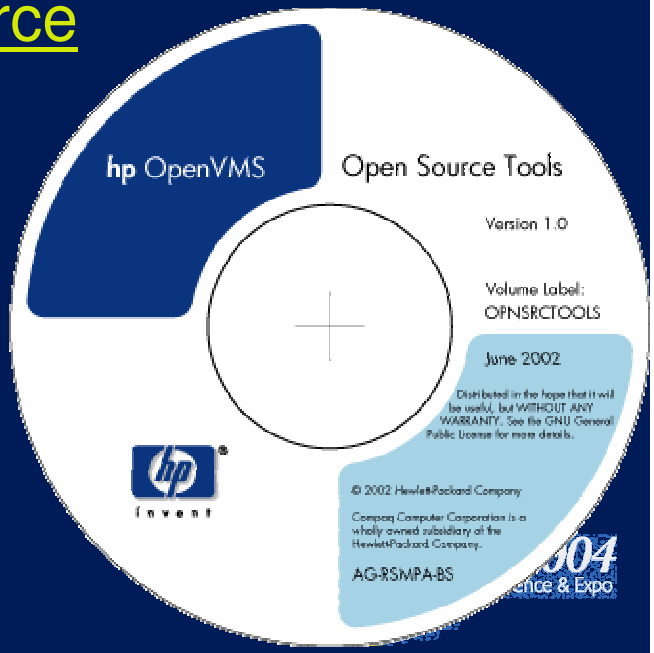
GNV

GNV

- GNU's not Unix
- GNV's not VMS
- BASH shell
  - Bourne Again Shell
  - Ported from OpenSource
- Shell Utilities
  - Simple commands: cat, ls, rm,
  - Application development utilities:
    - make
    - gawk
- Unix (like) environment
  - Root directory
  - Unix-style file specs

Finding GNV

- GNV project website
  - http://gnv.sourceforge.net
- OpenVMS website – OpenSource page
  - http://h71000.www7.hp.com/opensource
- OpenVMS kit – OpenSource CD
  - OpenVMS V7.3-1; V7.3-2…



Installing GNV

- PCSI product installation
  - Run the `PCSI-DCX_AXPEXE` file then:

```
$ PRODUCT INSTALL DEC-AXPVMS-GNV-V0105-004-1.PCSI
```

- Requires ODS-5 disk
  - If not system disk, specify target at install time:

```
$ PRODUCT INSTALL /DESTINATION=ddccuu: GNV
```

- Add startup file sys$startup:systartup_vms.com
  - `sys$startup:gnv$startup.com`
- Optionally, add login file to sys$manager:sylogin.com
  - `gnu:[lib]gnv_setup.com`
  - Or add it to your own login.com

Why ODS-5?

- Enables Unix-style filenaming
  - Also compatible with Microsoft Windows
- Filenames with funny characters, multiple dots

```
tar-1.13.25.tar.gz
This,Is@a#funny$filename%Dot.txt
```

- Directories with multiple dots
- Deep directories
- Optionally implements
  - Hard links
  - File access dates

Configuring

- User and/or data disks should also be ODS-5
  - **`SET VOLUME ddnnn:/STRUCTURE=5`**
- Enable hardlinks
  - **`SET VOLUME ddnnn:/VOLUME_CHAR=HARDLINKS`**
  - Not always necessary, but recommended
  - Required for some CONFIGURE scripts

GNV tips and tricks

- **`$ SET PROCESS/PARSE_STYLE=EXTENDED`**
  - Enables preservation of case in DCL commands
  - Also affects BASH
  - You'll need this for running CONFIGURE scripts

- **`$ SET PROCESS/CASE=SENSITIVE`**
  - If you really need it
  - Not generally needed
  - Enables full case sensitivity of file names
  - I don't recommend this, unless you need it
  - Some VMS applications won't take mixed/lower case files

BASH Shell

- A shell is a CLI (Command Language Interpreter)
- BASH is the Bourne Again SHell
  - after Stephen Bourne who wrote the Bourne shell
- Case sensitive (even if filenames aren't)
- Variables
- Conditionals (if… then… else…)
- Looping constructs (for, while, until)
- Program execution (no RUN command… just name the file to run)
- There is a BASH Reference Manual in the GNV kit
  - Its BASH V2. Software is V1.
  - Also, search the web for documentation.

Introduction To BASH

Hands On Training!

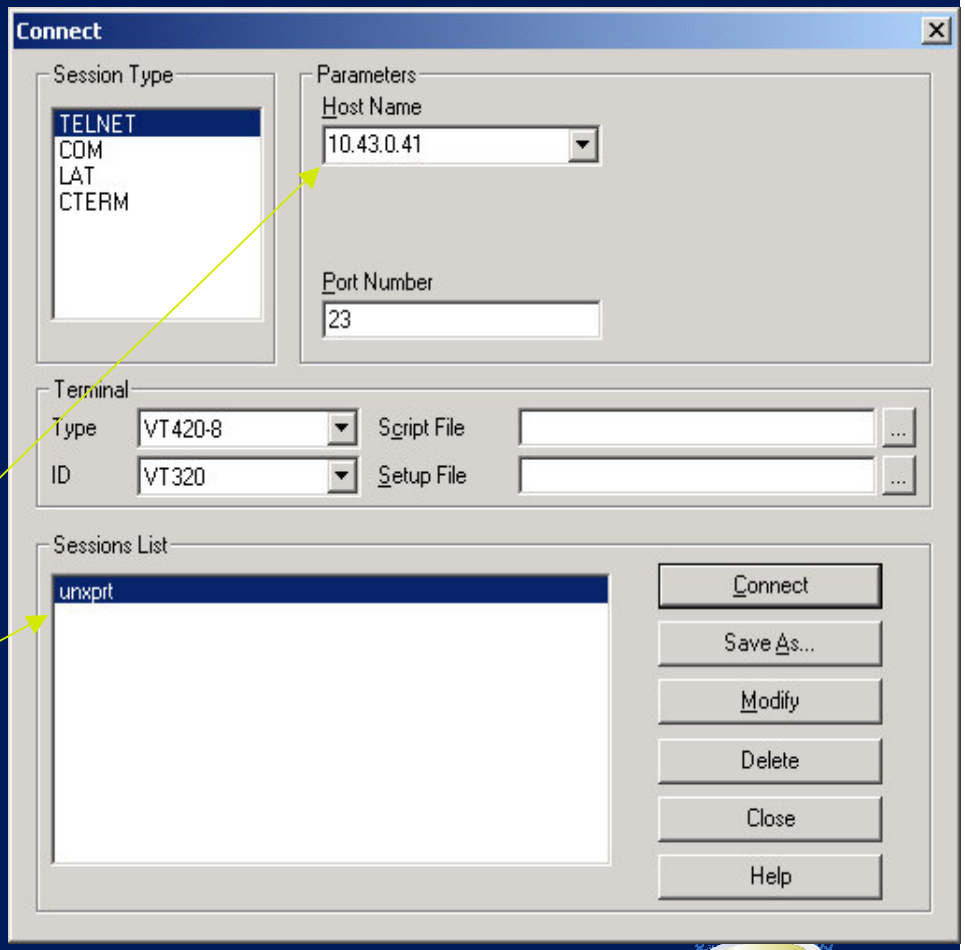
“Hands On” Lab

- Following slides designed as an instructor lead lab exercise.
- Provides a very high level introduction to some basic BASH functionality.
- Those with UNIX/Linux/BASH experience are welcome to follow along, or, explore the BASH shell on their own.

Logging In & Connecting

- Login to Windows
  - Accounts: administrator
    Passwords: <no password>
- Start PowerTerm
  - Programs->PATHWORKS PowerTerm 525->PowerTerm
  - Connect: Communication->Connect
    - Telnet to 10.43.0.41

    Or
    - Select UNXPRT



OpenVMS Lab accounts

- Accounts: user1… user10
- Passwords: user1… user10

- Set the terminal & process attributes
  - **`SET TERM /INQUIRE`**
  - **`SET PROCESS/PARSE=EXTENDED`**
- Login directories should exist, and be empty.

- Customize any way you wish
  - Feel free to create a login.com – This is your account for the rest of the class time

BASH – Introductory Commands

- Start BASH

```
$ bash
bash$
```

- $ HELP
  - More on “man” and help later

```
$ bash man
What manual page do you want?
bash$
```

- Exiting BASH
  - exit
  - ^D (crtl/d)
  - Not ^Z

```
bash$ exit
exit
$
```

BASH - Command Options

- Like DCL command qualifiers
- Generally preceded by a single hyphen (“-”)
  - Single letter
  - Multiple options may be specified
  - e.g.

```
ls -ar
```

- Or preceded by double hyphen (“--”)
  - Word
  - Single option each
  - e.g.

```
ls --all --reverse
```

BASH - HELP

- Where's the HELP command?
- Most commands support either -h or - -help

  `ls --help`

  - Give brief synopsis… Mostly good as a reminder of options
- Most commands have manual pages

  `man ls`

  - Detailed documentation
- man invokes "less"
  - Scroll with arrow buttons
  - Use "q" to quit.

BASH – Introduction

- $ SHO DEFAULT
  - pwd

```
bash$ pwd
/SYS$SYSDEVICE/USERS/USERX
bash$
```

- $ DIRECTORY
  - ls
  - Empty directory ➔

```
bash$ ls
bash$
```

- If directory is empty, create an empty file
  - touch
- Execute the “ls” command again

```
bash$ touch file.txt
bash$ ls
file.txt
bash$
```

BASH – Introduction (cont)

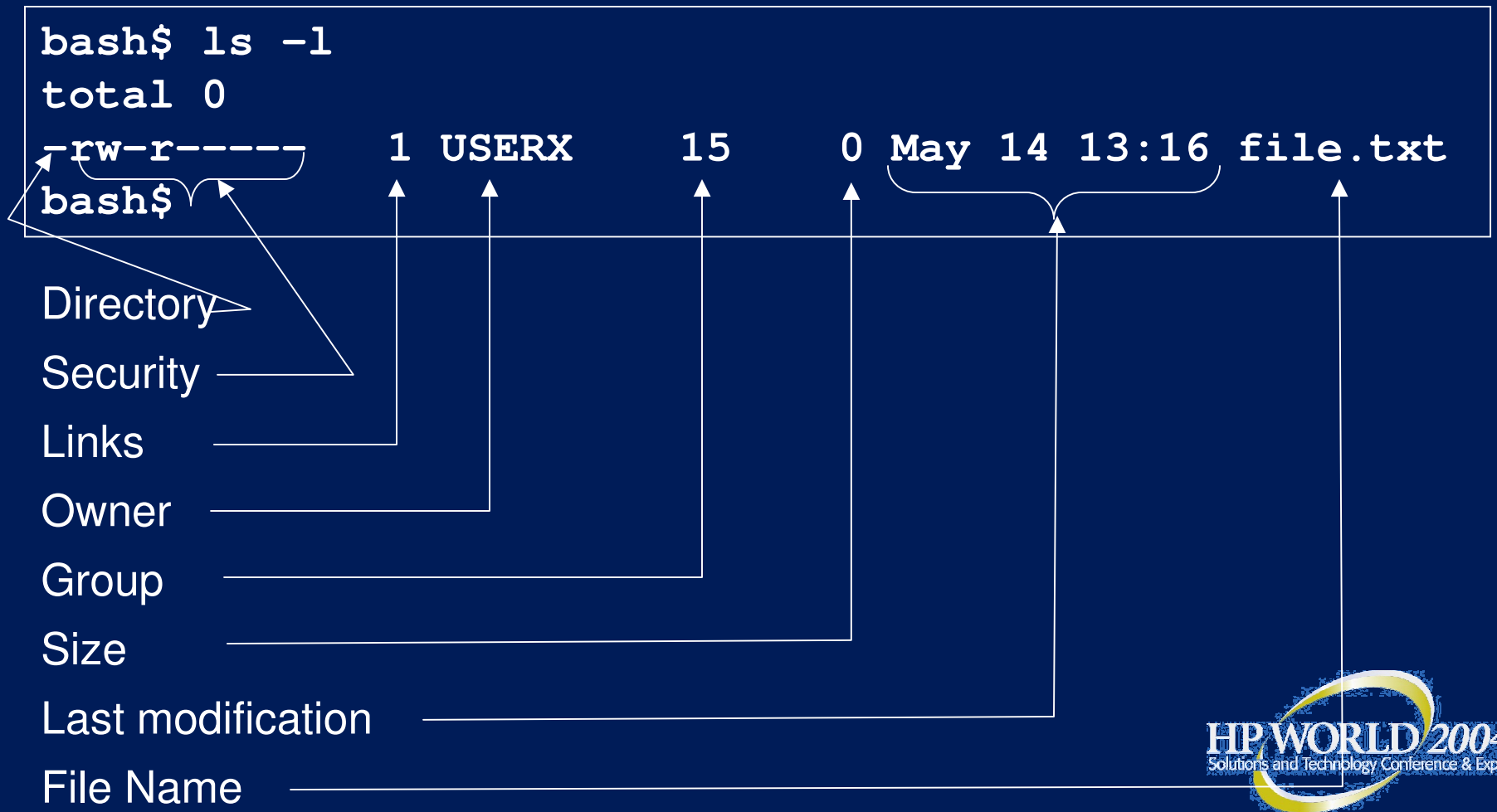
- $ DIR/DATE/SIZE/OWN/PROT
  - ls -l

```
bash$ ls -l
total 0
-rw-r-----    1 USERX    15     0 May 14 13:16 file.txt
bash$
```

- How do I interpret this output?
  - See next slide

BASH – Introduction (cont)

- $ DIR/DATE/SIZE/OWN/PROT



BASH – Introduction (cont)

- $ CREATE/DIRECTORY [.WORK]

```
bash$ mkdir work
bash$
```

- Execute ls –l again to see the directory:

```
bash$ ls -l
total 0
-rw-r-----    1 USERX     15       0 May 14 13:16 file.txt
drwxr-x---    1 USERX     15     512 May 14 15:04 work
bash$
```

Notice directory indication now?

Interacting with VMS
Dealing with “$” in BASH

- Using VMS logical names, for example sys$login:

```
bash$ ls sys$login
ls: sys: no such file or directory
bash$
```

- $ character indicates variable substitution - Needs to be escape encoded – try this:

```
bash$ ls sys\$login
ls: sys$login: no such file or directory
bash$
```

- Getting closer – now it is looking for file or directory named “sys$login” – How do we get it to resolve the logical?

Interacting with VMS - logical names

- VMS path: DEV:[DIR.SUBDIR]FILE.EXT
  - Roughly equivalent to:
- UNIX path: /dev/dir/subdir/file.ext
- Logicals need to be the “device” to get interpreted:
  - “SYS$LOGIN” would be “/SYS\$LOGIN”:

```
bash$ ls /sys\$login
file.txt  work
bash$
```

Interacting with VMS - logical names

- Kits used in future exercises are in **SYS$SYSDEVICE:[COMMON.KITS]** – how do we represent that in bash?

```
bash$ ls /sys\$sysdevice/common/kits/
barcode-0.98.tar.gz  cpio-2.5.tar.gz  m4-1.4.tar.gz
bash$
```

Interacting with VMS

- Try some other VMS directories
  - e.g. your login device and directory

```
bash$ ls /UNXPRT\$DKA0/users/userx
file.txt  work
bash$
```

Interacting with VMS

- Can still use DCL!
  - BASH will pass unrecognized commands to DCL

```
bash$ dir UNXPRT\$DKA0:[USERS.USERX]

Directory UNXPRT$DKA0:[USERS.USERX]

.bash_history;1     file.txt;1       work.DIR;1

Total o 3 files.
bash$
```

Moving around

- Changing directory

- $ SET DEFAULT [.WORK]

```
bash$ cd work
bash$ pwd
/SYS$SYSDEVICE/USERS/USERX/work
bash$
```

Moving around (cont.)

- Special Directory names:

VMS		BASH	
– []	←→	.	'dot' - Current directory
– [-]	←→	..	'dot dot' - Parent of current directory

- To move up a directory ($SET DEF [-])
  - cd ..

```
bash$ pwd
/SYS$SYSDEVICE/USERS/USERX/work
bash$ cd ..
bash$ pwd
/SYS$SYSDEVICE/USERS/USERX
```

File Manipulation

- **$COPY ← → cp**
  - COPY FILE.TXT COPY-OF-FILE.TXT

```
bash$ cp file.txt copy-of-file.txt
bash$ ls
copy-of-file.txt   file.txt   work
bash$
```

- **$RENAME ← → mv**
  - $RENAME COPY-OF-FILE.TXT FILE2.TXT

```
bash$ mv copy-of-file.txt file2.txt
bash$ ls
file.txt   file2.txt   work
```

More File Manipulation

- **$DELETE ← → rm**
  - $DELETE FILE2.TXT

```
bash$ rm file2.txt
bash$ ls
file.txt  work
bash$
```

- Be careful – rm deletes all versions of the file.

File Versions

- UNIX does not have file versions.

- BASH
  - Runs on VMS, therefore files do have versions, and multiple versions are created.
  - BASH does not display multiple versions.
  - rm ($DELETE) will delete all versions of the file.
    - Assuming you have VMS delete permission to the files.
  - mv ($RENAME) only affects highest version.

Working with file versions

- Create 3 ‘versions’ of a file:

```
bash$ cat > foo.bar
file foo.bar version 1 <CTRL/Z>
bash$ cat > foo.bar
file foo.bar version 2 <CTRL/Z>
bash$ cat > foo.bar
file foo.bar version 3 <CTRL/Z>
bash$
```

- What did we create? 1 file, or 3?

```
bash$ ls foo.bar
foo.bar
bash$
```

Working with file versions (cont)

- Appears to be just the last version created:

```
bash$ cat foo.bar
file foo.bar version 3
bash$
```

- There really is 3 versions:

```
bash$ DCL DIR foo.bar

Directory SYS$SYSDEVICE:[USERS.USERX]

foo.bar;3          foo.bar;2          foo.bar;1

Total of 3 files.
bash$
```

Working with file versions (cont)

- mv – Notice it does not rename all versions:

```
mv foo.bar foo.new
bash$ ls foo.*
foo.bar  foo.new
bash$ cat foo.bar
file foo.bar version 2
bash$ DCL DIR foo.bar

Directory SYS$SYSDEVICE:[USERS.USERX]

foo.bar;2                 foo.bar;1

Total of 2 files.
bash$
```

Working with file versions (cont)

- rm – Does delete all versions

```
bash$ rm foo.bar
bash$
bash$ ls foo.*
foo.new
```

Editors

- vi
  - GNV ships VITPU
    - TPU program that looks like vi
    - Warning: To exit:
      - To quit: `:q!`
      - To write output and exit: **`<ESC>ZZ`**
    - **`<ESC>`** is always a good idea
    - Not going to teach vi today!
- TPU/EDT
  - GNV is still VMS. You can always use VMS utilities

```
bash$ edit /tpu file.txt
```

BASH Introduction

- Use your favorite editor add contents to file.txt
  - I added 6 lines of text.

- Type the file
  - **$TYPE ← → cat**

```
bash$ cat file.txt
Hello OpenVMS Bootcamp -
Line2 - Second line of text
Line3 - Third line of text
Line4 - Even more text
This is the 5th line of text.
Last Line of text - Good Bye!
bash$
```

BASH Introduction

- more
  - Utilitity for screen by screen output
    - User advances output by hitting space bar
  - On GNV/BASH – ‘more’ is ‘less’
    - less and more are similar utilities, but, one and the same in GNV
    - less has added capabilities

```
bash$ more file.txt
Hello OpenVMS Bootcamp -
Line2 - Second line of text
Line3 - Third line of text
Line4 - Even more text
This is the 5th line of text.
Last Line of text - Good Bye!
file.txt (END)
```

Utilities Comparison

$ DIR	bash$ ls
$ DIR/SIZ/DAT/SEC	bash$ ls -l
$ TYPE	bash$ cat
$ TYPE/PAGE	bash$ less
$ DELETE	bash$ rm
$ DELETE […]*.*;*	bash$ rm -rf *
$ SHOW SYSTEM	bash$ ps -a
$ RUN foo	bash$ foo
$ SEARCH file.txt string	bash$ grep string file.txt
$ DIR […]foo*.txt	bash$ find ./ -name foo*.txt

Intermediate BASH

Intermediate Level BASH

- The next slides discuss various topics that might be useful to know.

- It is expected that the student will explore these topics as desired

BASH shell (continued)

- Redirection of input and output

```
ls –l > listing.txt
```

  - Output of “ls” command goes to the file “listing.txt”

```
cat > test.dat
```

  - Since no input specified, copies input stream (keyboard) to the named file. Until you hit ^Z. (On Unix, it would be ^D.)

```
cat > test.dat >>eof
Some data
Some more data
eof
```

  - Copy input stream, until “eof”. Works only in scripts.

BASH shell (continued)

- Pipes
  - `ls -l | sort -k 1,30`
    - Gives you a directory listing, sorted by file size
  - `ls -l | less`
    - Displays a long directory listing page by page
  - `grep "string" file.txt | less`
    - Searches file.txt for the string
  - `ls -l | grep "file.txt"`
    - Display just "file.txt"
    - Equiv: `ls -l file.txt`

BASH shell (continued)

- Command Line Completion
  - Type first few characters of a filename
  - Hit tab
  - BASH completes as much as is unique
  - Try:
    - `ls -l /gnu/lib/gn<tab>`
    - `ls -l /gnu/lib/GNV_`

    Type “S”:
    - `ls -l /gnu/lib/GNV_S<tab>`
    - `ls -l /gnu/lib/GNV_SETUP.COM`

BASH: Command line recall & history

- Up-arrow/Down-arrow just like DCL
- '!' to recall by number or string
  - Be careful - It is recall and execute!
- History preserved across sessions

```
bash$ history
    1  date
    2  sho time
    3  pwd
    4  cd ..
    5  ls
    6  rm -rf m4-1.4
    7  tar -xf m4-1.4.tar
    8  cd m4-1.4
    9  pwd
   10  cd ..
   11  sho time
   12  rm -rf m4-1.4
   13  tar -xf m4-1.4.tar
   14  tar -xf m4-1.4.tar
   15  history
bash$ !1     (recall by number)
date
Wed Nov  5 19:32:08 EST 2003
bash$ !pw    (recall by first characters)
pwd
/SYS$SYSDEVICE/USERS/USERX/WORK/M4
bash$
```

Bash

- Comment character

  #

- First line of most shell scripts

  #! /bin/sh

  – I believe this instructs exec() how to process the file

- Lists:

  – Space separated (No commas)

  ls /bin/ls /bin/rm /bin/make

- Shell processes wildcards

  ls *.txt

  – Generates:

  ls a.txt b.txt c.txt d.txt

Regular Expressions

- String expressions
  - Full of really fancy wildcarding and such
    - Lots of special characters
  - Used especially in SED scripts
- A really good resource:
  - http://sitescooper.org/tao_regexps.html

Shell commands from DCL

- Shell commands from DCL often work, but are unsupported
  - These all work.

```
$ less file.txt
$ ls
$ ls file.txt
```

  - Doesn't work. ls expects BASH to process wildcards

```
$ ls *.txt
```

- Use instead:

```
$ bash -c "ls *.txt"
$ bbb:==bash -c
$ bbb "ls *.txt"
```

Unix File Naming

Unix filespec

- Absolute filespec – Begins with a "/"

  `/directory/directory/filename`

  - Relative to the root
- Relative file spec – Doesn't begin with a "/"

  `filename`

  `directory/directory/filename`
- Filename may include any number of dots
  - E.g. filename.ext or tar-1.13.25.tar.gz
- Filenames may include almost any character
- Unix Root includes everything in the system
  - Disks; Devices; System directories; etc.
- Symbolic Links
  - Points to a file located elsewhere

OpenSource Software

OpenSource Software

- There's tons of OpenSource software
  - Part of the purpose of the UP initiative is to make it practical to run this on OpenVMS
- Go search
  - www.gnu.org
  - www.sourceforge.net
  - www.savannah.gnu.org
  - Lots more
- Most is designed to be installed on any variant of Unix

Installing OpenSource software

- Distributed in source form
- Contain configuration and build scripts
- Simply run the configuration script
  - It generates one or more makefiles
- Run the makefile
- This works on OpenVMS, under GNV, too.

Simple Open Source Porting Exercise

SYSTEM V BANNER

Instructor Lead Lab

- The set of slides are designed to be an instructor lead lab
- You are invited to follow along at your terminal.
  - If you encounter any problems, or, the instructor is moving to fast, please let us know!

SYSVBANNER

- Displays a horizontal banner in a manner similar to SYSTEM V
- Downloaded from http://packages.debian.org/stable/misc/sysvbanner
- Very simple exercise

SYSVBANNER (cont)

- Make a directory to work in – slides use "work/banner"
  - **`mkdir banner`**
- Change to the new directory
  - **`cd banner`**

```
bash$ pwd
/SYS$SYSDEVICE/USERS/USERX/work
bash$ mkdir banner
bash$ cd banner
bash$
```

SYSVBANNER (cont)

- Copy the zip file
  - `cp /sys\$sysdevice/common/kits/sysvbanner_1.0-10.tar.gz .`
- Check to see that it is there
  - `ls`

```
bash$ cp /sys\$sysdevice/common/kits/sysvbanner_1.0-10.tar.gz .
bash$ ls
sysvbanner_1.0-10.tar.gz
bash$
```

SYSVBANNER (cont)

- Uncompress it
  - bash$ gzip -d sysvbanner_1.0-10.tar.gz

```
bash$ gzip -d sysvbanner_1.0-10.tar.gz

gzip: sysvbanner_1.0-10.tar.gz: decompression OK,
      trailing garbage ignored
bash$
```

- Notice this message?
- Sometimes the record attributes aren't right when copying from the web

SYSVBANNER (cont)

- We will fix the zip file's attributes, so that it unzips without error
- Clean out anything that is in the banner directory
  - **`ls`** – to see what is there
  - **`rm`** – to remove it

```
bash$ ls
sysvbanner_1.0-10.tar
bash$ rm sysvbanner_1.0-10.tar
bash$
```

SYSVBANNER (cont)

- Copy the gz file again
- This time we will set the attributes on it
  - You will likely need to do this to each time you download a file from the web.

```
bash$ cp /sys\$sysdevice/common/kits/sysvbanner_1.0-10.tar.gz .
bash$ ls
sysvbanner_1.0-10.tar.gz
bash$
```

SYSVBANNER (cont)

- Set File Attributes
  - Use DCL command from within BASH!
    - Need to use VMS friendly file specification
    - Notice “^” characters

```
bash$ DCL SET FIL/ATT=RFM=STMLF sysvbanner_1^.0-10^.tar.gz
bash$
```

- Do the unzip again:

```
bash$ gzip -d sysvbanner_1.0-10.tar.gz
bash$
```

- Clean!!

SYSVBANNER (cont)

- What was un-zipped?
  - **`ls`**

```
bash$ ls
sysvbanner_1.0-10.tar
bash$
```

- A “tar” file (UNIX archive file)
- Next step is to expand it
  - **`tar -xf sysvbanner_1.0-10.tar`**

```
bash$ tar -xf sysvbanner_1.0-10.tar
bash$
```

SYSVBANNER (cont)

- What is created now?
  - `ls`

```
bash$ ls
sysvbanner-1.0  sysvbanner_1.0-10.tar
bash$
```

- Change to the newly created sysvbanner-1.0 directory
  - `cd sysvbanner-1.0`

```
bash$ cd sysvbanner-1.0
bash$ ls
Makefile  banner.1  banner.c  debian
bash$
```

SYSVBANNER (cont)

- What is in this new directory
  - `ls`

```
bash$ ls
Makefile  banner.1  banner.c  debian
bash$
```

- Looks like at least one source file
- Most important – a “Makefile”

SYSVBANNER (cont)

- Run Make
  - **`make`**
- You should see this:

```
cc -Wall -O2 banner.c -o banner
? cc: Unrecognized switch -Wall
```

  - first line tells us what it is doing
  - 2nd line can be ignored
- Should also see two "%CC-I-" informational messages – OK to ignore.

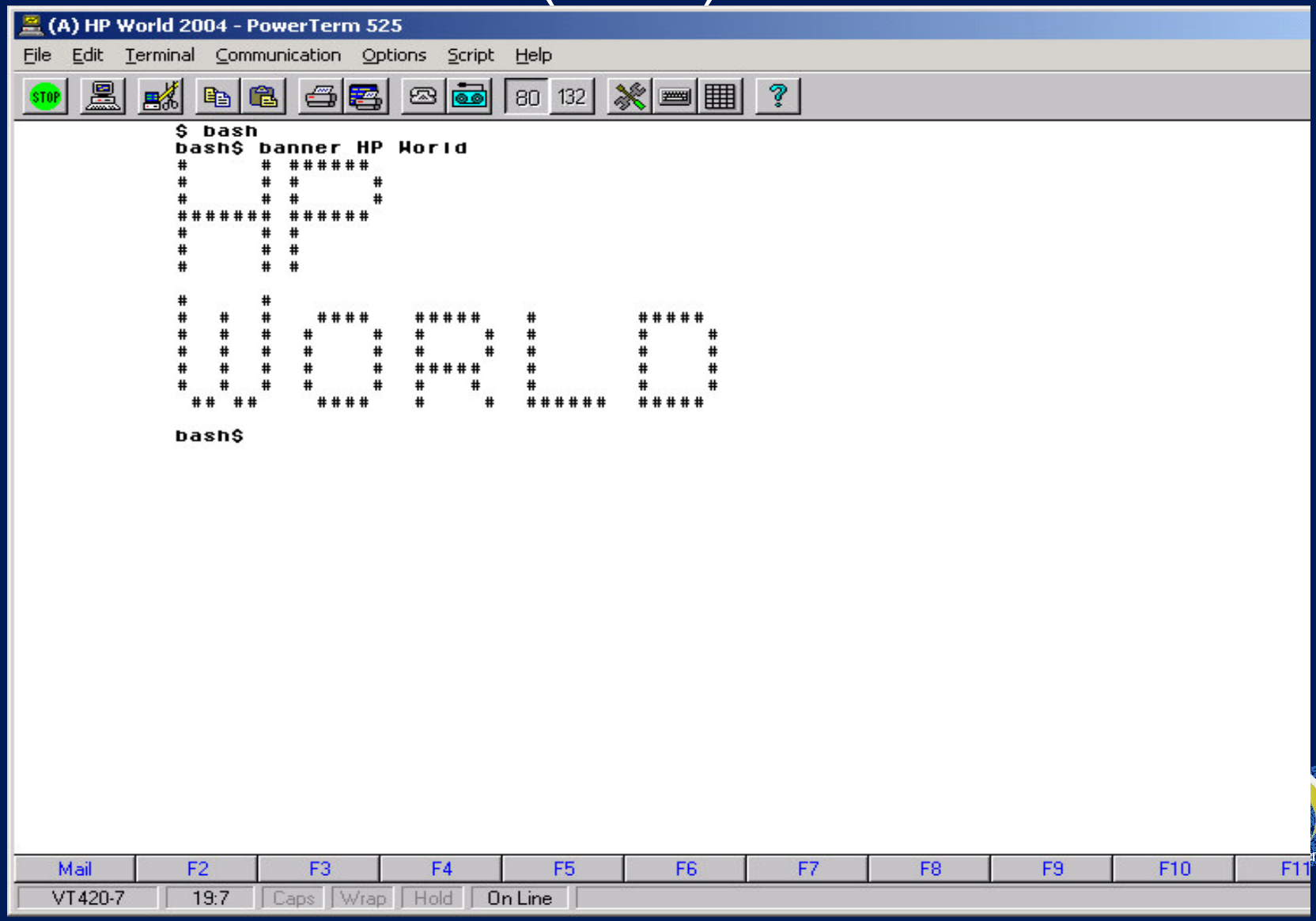
SYSVBANNER (cont)

- What did it do?
  - **`ls`** – to see what was created

```
bash$ ls
Makefile  banner  banner.1  banner.c  banner.o
debian
bash$
```

- Notice new file “banner” – likely, that is your program.
- Try Running it
  - **`bash$ banner HP World`**

SYSVBANNER (cont)



HP WORLD 2004

Solutions and Technology Conference & Expo

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional