

Chapter 3

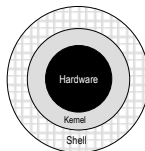
Getting Started

Overview

- ▣ What is a Shell?
- ▣ Running scripts
- ▣ The PATH environment variable
- ▣ Sub-shells


Lesson: What Is the Shell?

- ▣ Command execution
- ▣ Environment settings
- ▣ Variable assignment
- ▣ Variable substitution
- ▣ Command substitution
- ▣ Filename generation
- ▣ I/O redirection
- ▣ Pipelines
- ▣ Interpretive programming language

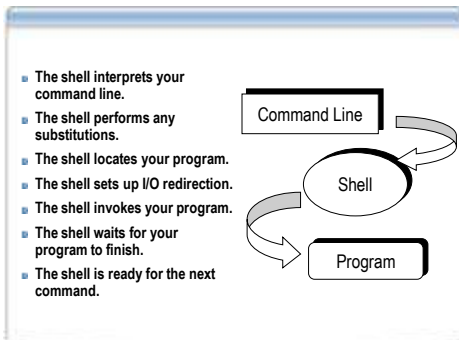


Commonly Used Shells

File	Description
/bin/sh	POSIX / Bourne shell
/bin/ksh	Korn shell
/bin/csh	C Shell
/bin/ash	Almquist Shell
/bin/bash	Bourne-Again shell



What Happens after "Enter?"



Lesson: Running scripts

- A shell program is a regular file containing UNIX system commands.
- The file's permissions must be at least „read“ and „execute“.
- To execute, type the name of the file at the shell prompt.
- Data can be passed into a shell program through
 - environment variables
 - command line arguments
 - user input

Additional Techniques

- Document shell programs by preceding a comment with a number sign (#).

```
sh shell_program arguments
```

- shell_program* does **not** have to be executable.
- shell_program* does have to be readable.

```
sh < shell_program
```

- Each line of *shell_program* is read and used as input from keyboard.

Which Shell Interprets the Script?

- First line of the script can decide, which program is used to run script

- # ! - „magic bang“

- Example

```
$ cat myscript.sh
#!/bin/sh

# Displays list of files
# in the current directory

echo " Current Directory"
echo "===== "
ls | pr -3t
```

Portability Issues

Korn family Shell	Bourne Shell
<code>\$(command)</code>	<code>`command`</code>
<code>function fname</code>	<code>fname ()</code>
<code>((a = a + 1))</code>	<code>a=`expr \$a+1`</code>

- There are compatibility and portability issues when deciding on which syntax to use.
- When reading scripts, do you understand the syntax used?



Lesson: The PATH environment variable

- A list of directories where the shell will search for the commands you type

```
$ env
PATH=/bin:/usr/bin:~/bin
```

- Working directory is not by default in path

```
$ ls
p1d
$ p1d
sh: p1d: not found
$ ./p1d
```

Lesson: Sub-shells

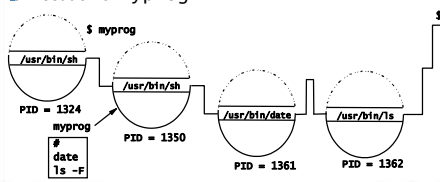
- A shell script may call another shell script.

```
$ cat ./first
# File: first
car1=ford
car2=chevy
echo "Parent car 1: $car1"
echo "Parent car 2: $car2"
./second # Run the second script.
sleep 1 # Keep echos sequential.
echo "Parent car 1: $car1"
echo "Parent car 2: $car2"
```


Example Shell Program

```
$ cat myprog
# this is program called myprog
date
ls -F
```

- Execution of myprog




Review Exercises



■ Complete the exercises from the Learning Guide

Topics for Review



- 1 Read the review topics
- 2 Think about what you learned in this Session in the context of your own work environment
- 3 Discuss your answers as a class
