**3**

**Redirection and Pipelines**

**Objectives**

·Explain the concepts of standard input, standard output and standard error

Redirect input/output of a command to or from a file

·Connect standard input to standard output via pipeline



**Notes**

·As characters are typed on the command line the Shell is scanning for special characters such as |,>, <, >>, or <<. Each of the characters prompts the Shell to perform a specific function.

**Pipeline Examples**

- **Symbol for a pipe**

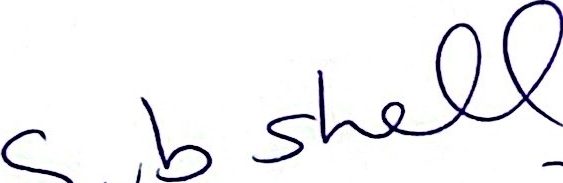
**(^ is an obsolete symbol for a pipe)**

$ 1s | pr -3 -115

**Causes the standard output of** 1s **to become standard input of** pr.

$ ps -ef | sort -n +1

**Sorts the current process table by PID.**



Notes

·Any command capable of making use of the standard input and output can appear in a pipeline. The command is called a filter. A filter reads from the standard input and writes to the standard output.

$(date ; ps -ef | wc -1) | pr -115

What output is generated by this pipe?

1s is not a filter because it does not read from standard input. If it is used in a pipe, it must be the first command in a pipe.

1pr (or 1p) is not a filter. It must be the last command if used in a pipeline.

**Helpful Hints Using Pipelines**

1. Decide on the correct command to **process the** **input** data.Sketch **pipeline.**

2. Typical commands for **the** **following** **actions:**

**counting:** WC

searching: grep

sorting: sort

removing repeated lines**:** uniq

**producing multicolumns:** pr

3. Use lists and semicolons.

4. To increase performance,put **filtering** commands that reduce **the** **amount** of data early in pipeline.

**Notes**

· The position of a command in a pipeline can be critical to performance.

Example:Suppose we have a huge file (100,000 lines) with names in it. We want a sorted list of all people named "Bil1".

Pipeline Comment

cat hugefile | sort | grep Bill Slow, we have to sort 100,00 lines.

cat hugefile | grep Bill | sort Better,weonlyhavetosort,say100lines.

**Differences Between Pipelines and**

**Redirection**

**·Two or more commands connected by a pipe constitute a pipline.**

**Pipeline** **Redirection**

$ 1s -1 | wc $ 1s -1 > file1

$ wc < file1

**Notes**

The results are equivalent; however, the user is responsible for creating filel and removing file1 when it is no longer of use in the system.

· DOS implements pipes using temporary files and redirection.

**Differences Between Pipelines and**

**Redirection (Cont'd)**

**· Example 1:**

$ echo "Output goes with errors" **2>file1** **1>&2**

**· Example 2:**

**What is the difference between the command above and the following** **command?**

$ echo "Output goes with errors" **1>&2** **2>file1**

**Memory Locations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | stdin 0 | stdout 1 | stderr 2 | file1 |
| Example 1 | tty | file1(copy  fromlfd 2) | file1 |  |
| Example 2 | tty | tty (copy  from fd 2) | file1 |  |



**Notes**

Redirection occurs left to right. In the first example file “file1" is created before stdout is redirected to the error file. Thus stdout goes into filel.

In the second example, stdout is redirected to stderr, which at that time is the terminal display.

**Summary**



Processes start out with three open files, standard input, standard output,and standard error. By default, they are connected to the terminal.



These files have associated file descriptor numbers, 0, 1, and 2 respectively.

The shell provides operators to change the files associated with these file descriptors.

> Change standard output

Create or truncate named file

>> Change standard output

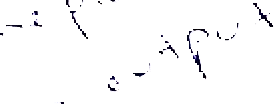
Append to named file

< Change standard input

Read from named file

Take input data from text of shell program

·Two file descriptors can share the same file. This is most often done to make standard error and standard output go to the same file.

**·Pipelines connect two running programs,** makingtheoutput of thefirst **become the input to the second.**

**Notes**

**Lab 3**

**Objective:**

You have seen how special symbols can be used to redirect standard input, standard output,and standard error to and from commands. You will run commands using these symbols and investigate the source and destination of the corresponding data.

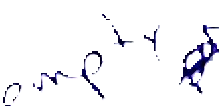
**Exercises:**

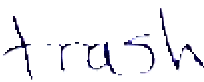
1. Use the “secret” directory from lab 2. Enter the command “cd" to make sure your current directory is your home directory.

Execute each of the commands below and determine the location of the standard output and standard error.

|  |  |  |
| --- | --- | --- |
| Command |  | Location |
| find . -name secret -print | stdout  stderr |  |
| find . -name secret -print 1>file1 2>file2 | stdout  stderr |  |
| find . -name secret -print >file3 | stdout |  |
| find. -name secret -print 1>file4 2>&1 | stdout  stderr |  |



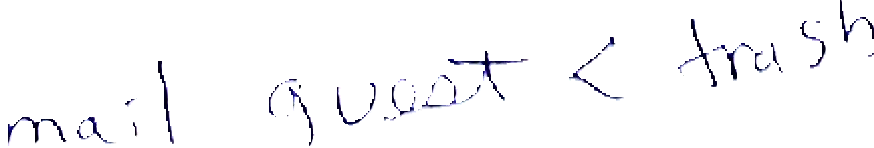
2.Create an empty file using redirection. 

Append a message to the file. 

Read the file usi gessecoo n 



3. Append the output of the date command to the file in #2.



Mail the file to yourself.

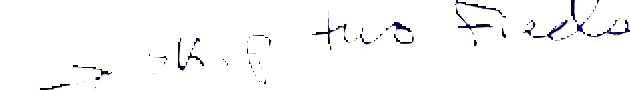
**Use a pipeline to gather the following information**



4.How many users are on your system now? (Hint: use who and wc)

5. Get a sorted list of the logged-on users on the network. (Hint: use rwho and sort)

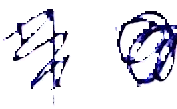
6. Eliminate duplicate names from the list in #5 (Hint: use uniq in your pipe)

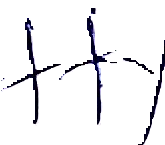
7.Find all processes associated with your login shell and sort according to PID number. (First use the tty command to identify your window or terminal).

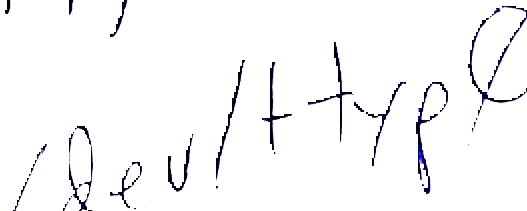
L

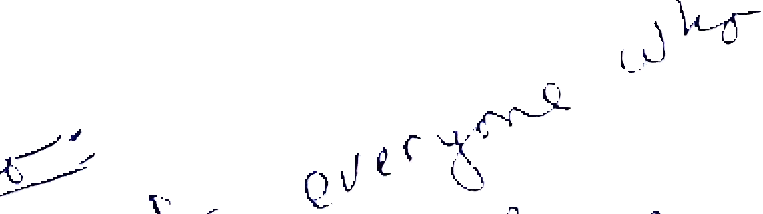
ps ax1 | grep <terminal> |sort +2n What was the PPID (Parent process-ID) of your login shell?

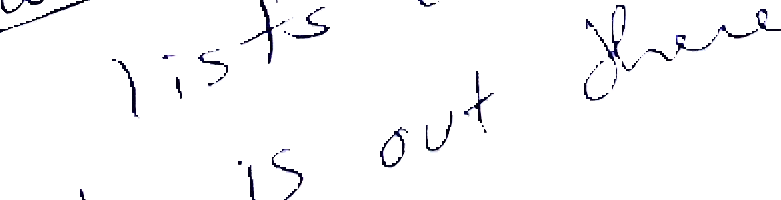
What was the PID of your login shell?

8. Count the number of files in your current directory. Do not include directories or symbolic links. 









UNIX Shell Programming 3-15

© Copyright 1996-2000 GTRC,All Rights Reserved

**4**

**Creating a Com** **mand File**

Objectives

· Create a Shell script or command file

·Make a Shell script executable

·Place the file in an appropriate location for execution

·Change the default file creation mode



**Notes**

· A shell script can be typed at the terminal and executed upon completion or it can be enter into a file, created using a text editor,and then executed.

**Command File**

·To create a *command* file, the following three tasks are necessary:

1.Create a file;

2.Make the file executable; and

3. Put the file in a directory where it can be found by the Shell.

**Notes**

·A simple-command is a sequence of non-blank words separated by blanks. The first word specifies the name of the command to be executed. It can be followed by options or arguments.

Shell commands can be grouped together in a *command* file.Whenever you need to repeat a long command or a sequence of commands, it may be advisable to group the commands in a *Shell script* or *command file.*

**File Creation**

**Files can be created in the following three ways:**

1. $ cat > file

2. $ > file Createsa file of **zero** length.

**The** **file** **commands** mv, cp, and In can be used to rename existing files,copy a file to another, or link a file with an existing file.

**3.** **Files** **can** be created by using one of the system editors.

**Notes**

cat reads from standard input and writes to file. Input continues until a Control-Dis received.

·cat > file either creates a new file or overwrites an existing file. If the file exists,the contents are overwritten; otherwise the file is created.

· > file is a quick way to create a file of zero length.

To use any of the three methods of creating files the user must have write permission in the directory.

**Filename Generation**

**Filenames** may be up to 255 characters in length.Any characters except the slash **and** **ASCII** NUL can be used. However, it is **best** **to** **avoid** the following **characters**:

<> & \* ? [ ] -

any nonprintable character

· Most filenames are composed of alphanumerics, underscores,and dot.

Notes

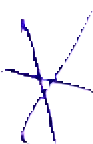
**Keywords Associated with File Creation**

**umask**

·umask **is used to eliminate default permissions when a** fileiscreated.

·Aumask **of** 077 **would shut off rwxfor group and other.**

**Most systems have a default file creation mode of** 666. **With a** umaskof **022,the new mode of creation is** 644.

**You can make a** **umask entry in your** .profilefile. 

**The** umask **value is inherited by all processes started by the Shell.**



**Notes**

· $ 1s -1 /etc/passwd

-rw-r--r--/etc/passwd

The first character gives the following information:

- file

d directory

C character device

b block device

p named pipe

1 symbolic link

S socket

The other nine characters refer to the file permissions:

Owner Group Other

rwx rwx rwx

UNIX Shell Programming

© Copyright 1996-2000 GTRC,All Rights Reserved

The values assigned to the permissions are as follows:

r read 4

W write 2

x execute 1

A value of 7 gives read, write, and execute permission.

The command:cc -o prog prog.c compiles the program prog.c and puts the output in prog.The compiler will add execution bits to the file permissions.

Example: $ umask 066

$ > status

$ 1s -1 status

-rw- status

C

**Execution of Shell Files**

**1.Make a** commandfile **executable using** chmod.

$ chmod 700 command.file **(absolute permission)**

**or**

$ chmod u+x command.file **(relative permission)**

**Now execute the** commandfile **by giving the name:**

$ command.file

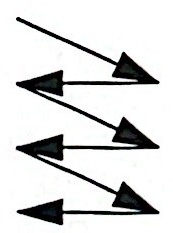
**2. Fork a subshell using** sh.

$ sh command.file

**Example:**

**PID** **PPID**

init 1

sh 123 1

sh 130 123

ps 145 130

**Notes**

**Other Ways of Executing Shell Files**

**·“.”executes a Shell script as part of the current process without spawning a** **new process. Changes made to variables after a dot command terminates** **are still in effect.**

$ cat current

echo "current process begins execution"

. command.file

echo "execution of current process continues"

exec **substitutes another process in place of the current process.**

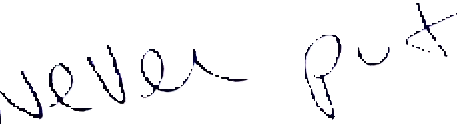
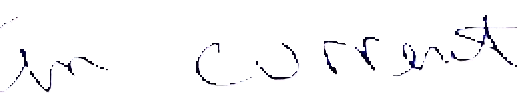
$ cat sub

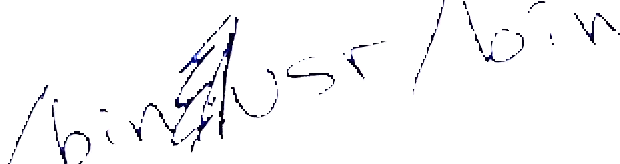
echo "current process begins execution"

exec command.file

echo "what happened to the original process?"

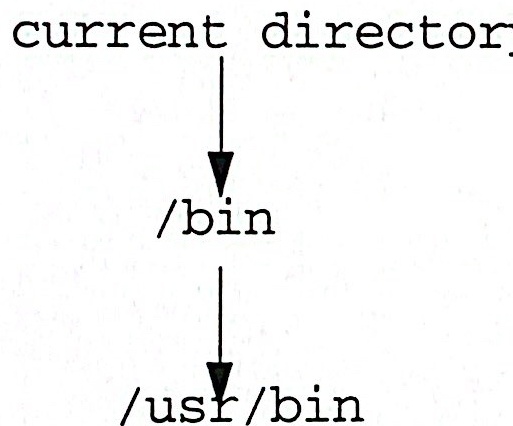
**Notes**

  Creating a Command File

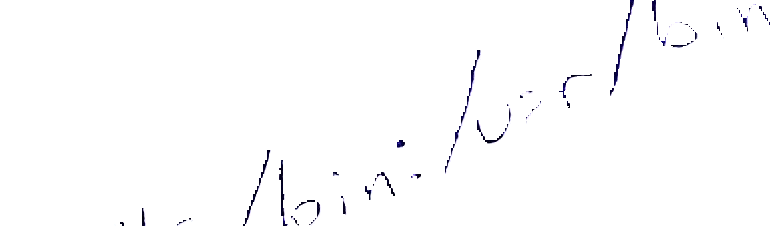
 **PATH Variable**

**Each user has a PATH variable. The directories listed in the PATH are the** **locations that the Shell looks to find executable programs, such as UNIX** **utilities or user defined scripts. If the default search path is:**

PATH=.:/bin:/usr/bin

**The shell looks through the following structure:**

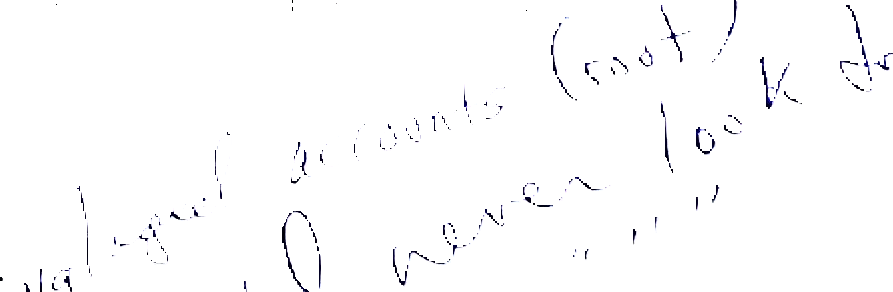
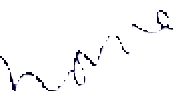
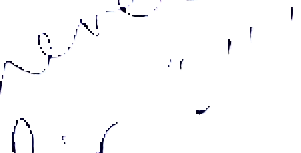




**Notes**

If the file is not found in one of the directories, you will receive an error message, such as:Command not found.

Each directory in PATH is separated by a:.

lageel accounts ( 

· ".”dot represents the current directory.

You can change the search order:

PATH=/bin:/usr/bin:.

You can also add other directories to the PATH.

PATH=/bin:/usr/bin:.:/usr/local/bin

The shell searches your PATH from left to right executing the first match it finds.

./ command means execute command in current directory.



**Creating an Executable Shell File**

**·In your HOME directory, create the file info including the** date **and** who **commands.**

$ cd

$ echo date > info

$ echo who >> info

**·Display the permissions associated with the info file.**

**What is the result if you type** info?

**·How can you correct the situation?**

**Type:** $ 1s -1 info

$ chmod u+x info

$ 1s -1 info

$ info

**Notes**

· A Shell file consists of one or more UNIX commands that can perform a specific task. The command file iscreated by entering the editor.

**The Shell Must Be Able to Locate An**

**Executable File**

$ cat $HOME/bin/findme; 1s -1 $HOME/bin/findme

echo Can you find me?

-rwxrw-rw- 1 team1 100 22 Jan 2 9:44 /users/team1/bin/findme

$ findme

ksh: findme: not found

$ which findme

$ echo $PATH

/bin/posix:/bin:/usr/bin:/usr/lib:.

$ PATH="$PATH:$HOME/bin"

$ echo $PATH

/bin/posix:/bin:/usr/bin:/usr/lib:.:/users/team1/bin

$ which findme

/users/team1/bin/findme

$ findme

Can you find me?

**Notes** 

The shell looks in the directories listed in the PATH variable to locate a file that you are trying to execute.

To add directories to a PATH variable, assign the PATH variable to contain the old PATH variable followed by any new directories.

 comma 