**Table Of Contents**

1 The Shell as a Process

Objectives....................................................................1-2

Kernel, Utilities, and Shell.....................................................1-3

The.............Shell....................................................................1-4Shell

History..................................................................1-6

How Does Your System Know What Shell to Start forYou?.......................1-7

Shell Startup..................................................................1-8

Login Sequence..............................................................1-9

Processes Tables.............................................................1-10

Shell Questions and Answers..................................................1-11

Excursion:UNIX System Calls...............................................1-12

Starting a Process in C........................................................1-13

Running a Program..........................................................1-14

Who Opens the Files?........................................................1-15

Redirecting IO..............................................................1-16

Shell Process...............................................................1-17

Foreground/Background Processing............................................1-18

Additional Shell Responsibilities..............................................1-19

Special Shell Features........................................................1-20

Examples of Features of the Shell.............................................1-21

Summary...................................................................1-22

Lab 1........................................................................1-24

2 Built-In Commands

UNIX Shell Programming..............1

©Copyright 1996-2000 GTRC,All Rights Reserved

Objectives...................................................................2-2

Built-In Commands...........................................................2-3

Advantages/Disadvantages of Built-in Commands...............................2-20

Summary...................................................................2-21

Lab 2.......................................................................2-22

3....Redirection and Pipelines

Objectives....................................................................3-2

Redirection(>,>>,<)........................................................3-3

File Descriptors..............................................................3-4

Standard Input, Standard Output, and Standard Error..............................3-5

Input/Output Redirection......................................................3-6

Pipelines:Connecting Command Processes.....................................3-8

Differences Between Pipelines and Redirection.................................3-11

Summary....................................................................3-13

Lab 3.......................................................................3-14

4 Creating a Command File

Objectives..............................................................4-2

Command File............................................................4-3

File Creation,.............................................................4-4

Filename Generation.......................................................4-5

Keywords Associated with File Creation......................................4-6

Execution of Shell Fles....................................................4-8

PATH Variable..........................................................4-10

Creating an Executable Shell File..........................................4-11

The Shell Must Be Able to Locate An Executable File.........................4-12

Summary................................................................4-13

UNIX Shell Programming..............1

©Copyright 1996-2000 GTRC,All Rights Reserved

5 Variables

Objectives.............................................................5-2

Definition of a Shell Variable.............................................5-3

Assigning Variables.....................................................5-4

The Values of Shell Variables.............................................5-7

Variable Assignments...................................................5-8

Predefined Shell Variables................................................5-9

-profile File...........................................................5-11

Initial Environment....................................................5-14

Shell Defined Variables (readonly)........................................5-16

Summary.............................................................5-17

Lab5................................................................5-18

6 Parameters

Objectives.............................................................6-2

Positional and Special Parameters to the Shell.................................6-3

Shifting Positional Parameters............................................6-5

Parameter Substitution...................................................6-7

Summary.............................................................6-11

Lab 6................................................................6-12

ton Cha

7 Pattern Matching

Objectives.............................................................7-2

Shell File Name Substitution Characters.....................................7-3

Shell Patter Matching Examples..........................................7-4

Quoting...............................................................7-5

UNIX Shell Programming..............1I

Copyright 1996-2000 GTRC,All Rights Reserved

Regular Expression Special Characters.......................................7-7

Utility Regular Expressions.................................................7-8

Evaluate an Expression....................................................7-10

Notes on expr............................................................7-11

Summary.........................................................7-14

Lab 7ghr h..................................................7-15

8 More on Commands

Objectives................................................................8-2

Command Line Evaluation..................................................8-3

Shell Script...............................................................8-7

Short-Circuit Operators....................................................8-10

Braces and Parentheses....................................................8-11

Summary................................................................8-12

Lab 8...................................................................8-13

9 Exit/Test

Objectives................................................................9-2

exit Status................................................................9-3

The test Command........................................................9-5

Types of Tests............................................................9-6

Tests on Numerical Values..................................................9-7

Tests on File Types.......................................................9-9

Tests on Character Strings.................................................9-12

Comparison of Character Strings and Numeric Strings..........................9-15

Logical AND and OR Operators............................................9-16

Summary................................................................9-18

UNIX Shell Programming..............1I

Copyright 1996-2000 GTRC,All Rights Reserved

Objectives................................................................10-2

Bourne Shell Programming..................................................10-3

Programming Concepts.....................................................10-4

Advantages/Disadvantages of Shell Scripts....................................10-5

Flow Control -if.........................................................10-6

Flow Control -elif........................................................10-9

Flow Control-case.......................................................10-11

Summary................................................................10-16

Lab 9 and 10.............................................................10-17

11 Looping Constructs

Objectives...............................................................11-2

Flow Control -for in.......................................................11-3

Flow Control-while.......................................................11-5

Comparison Between for and while..........................................11-8

Flow Control-until......................................................11-10

Programming Notes.......................................................11-13

Summary................................................................11-16

Lab 11..................................................................11-17

12 Here Documents

Objectives................................................................12-2

here Document.................................................................12-3

Summary......................................................................12-7

Lab 12........................................................................**12-8**

13 Exporting...........UNIX Shell Programming..............III

© Copyright 1996-2000 GTRC,All Rights Reserved

Objectives..................................................................13-2

Variables in **Processes.**......................................................**13-3**

Exporting Variables.........................................................13-7

Exporting Notes............................................................13-8

Summary..................................................................13-11

Lab 13....................................................................13-12

14 ErrorDebugging

Objective...................................................................14-2

Termination of a Shell Program..............................................14-3

For Debugging.............................................................14-4

Execution Flags............................................................14-5

Invocation Flags............................................................14-6

Error Handling.............................................................**14-7**

Signals.....................................................................**14-8**

Traps......................................................................14-9

Summary..................................................................**14-11**

Lab 14....................................................................14-12

15 Shell Functions

Objectives..................................................................15-2

Shell Functions............................................................15-3

Defining a Function.........................................................15-4

Examples..................................................................15-5

Lab 15.....................................................................15-6

16 Bourne-Again Shell

Objectives.................................................................16-2

UNIX Shell Programming IV

Copyright 1996-2000 GTRC, All Rights Reserved

Bourne-Again Shell...........................................................16-3

Command Line Editing........................................................16-4

Command History.............................................................16-6

Using Previous Commands.....................................................16-7

Built-in Integer Arithmetic...................................................16-10

Parameter Substitution.......................................................16-13

alias Command............................................................16-14

Job Control..................................................................16-15

Starting and Stopping Jobs.....................................................16-16

cd Command................................................................16-17

Tilde Substitution...........................................................16-18

Command Substitution........................................................16-19

Shell Variables...............................................................16-20

Quoting...................................................................16-21

Exported Variables..........................................................16-23

Select Command...........................................................16-24

Summary..................................................................16-25

Lab 16....................................................................16-26

Appendix A Exercises Solutions....................................................A-1

Appendix B Bourne-Again Shell Commands........................................B-1

Appendix C Typing Tutorial.......................................................C-1

INDEX.............................................................................1-1

**The Shell as a Process**

**Objectives**

·Define the function of the Shell

· Understand the login sequence in terms of fork and exec

·Be able to execute processes in the background

UNIX Shell Programming 1-1

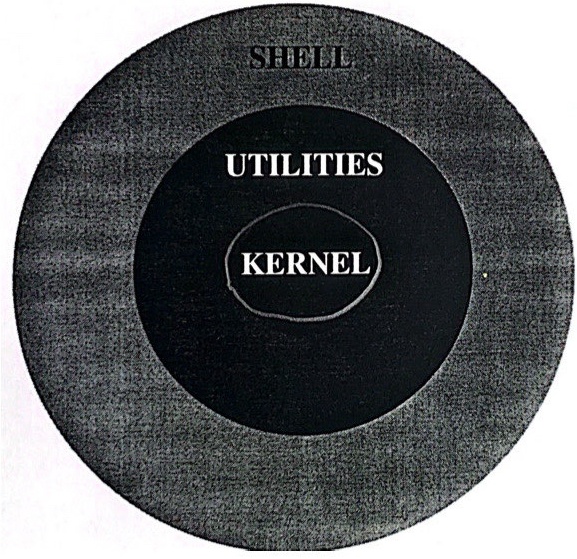
Copyright 1996-2000 GTRC, All Rights Reserved

·Analyze and interpet information on the command line

·Use exec to activate processes

**Notes**

**Kernel, Utilities,and Shell**



**Notes**

Kernel Communicates with disks,tapes,etc.

Manages computer's resources.

- Organizes the file system.

- Provides support for the utility programs.

Utilities Reside on a disk. 1s,ps,etc,

Bring into memory when command is executed.

Shell

Forms the interface between the users and the UNIX Operating System.

- Reads and parses command line data.

UNIX Shell Programming

1-3

**The Shell**

What is the Shell and what does it do?

- It is an ordinary user program.

It reads and interprets UNIX commands and issues appropriate UNIX system calls.

It expands file-matching wild-cards.

It performs redirection.

- It passes arguments to programs.

- It is similar to a programming language.

- It is written in the C language.



Notes

The Shell is an ordinary C Program.The Shell acts an intermediary between the user and the Kernel. It executes commands read from a terminal or a file.

·Q:How is the Shell similar to a programming language?

A:The Shell has variables, branching and looping constructs.

UNIX Shell Programming



**Which Shell**

Developed at AT&T Bell Laboratories by Stephen Bourne.

ksh *Korn Shell*

Developed at AT&T by David Korn.

**It is an extension of the sh Shell.**

csh ***C Shell***

**Developed at the University of California at Berkeley.**

**The syntax is similar to the C programming** language.

bash ***Bourne-Again Shell***

**Developed to provide an open source shell for Linux.**

**Based on sh, it contains features from ksh and csh.**

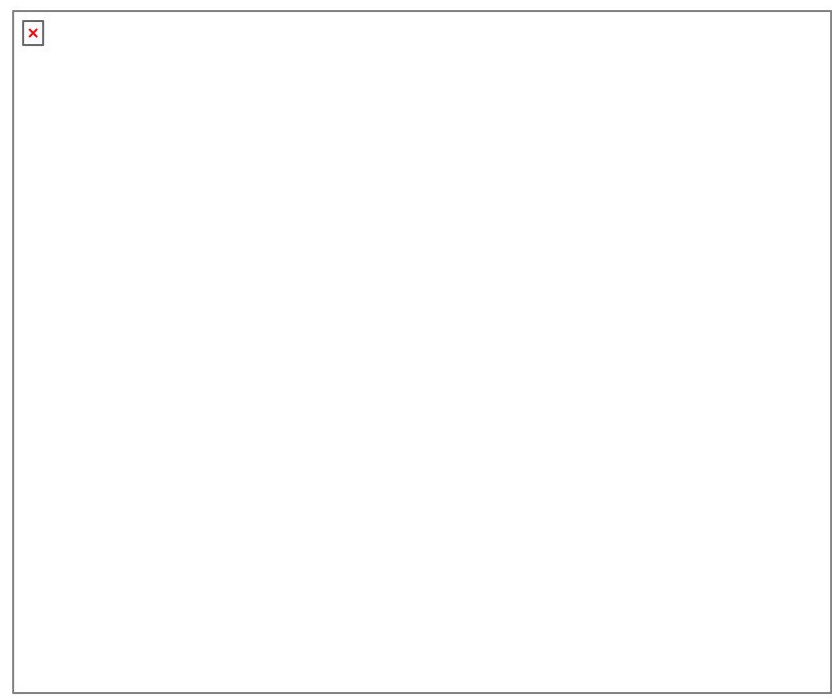
**Notes**

The Shell provides an interface between the user and the Kernel.Currently there are three Shells and you can select which one best meets your needs.

Some systems provide rsh,a restricted version of sh. The Korn Shell has a -r flag to provide the same functionality.

On other systems, rsh is the remote shell for executing commands on a system over a network.Those systems where rsh is the restricted shell often provide remsh for the remote shell.

**Shell History**



**Notes**

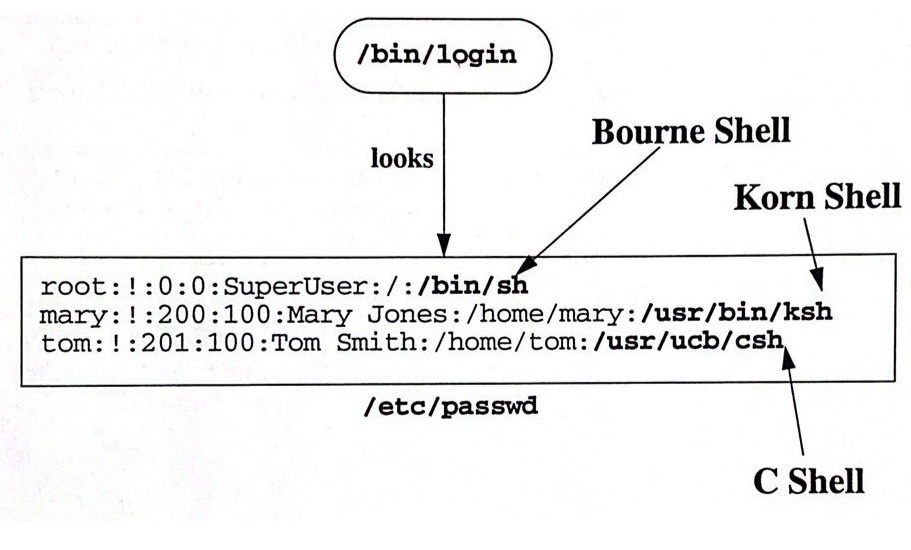
The Korn shell is larger than the C shell or the Bourne shell.

The Korn shell is compatible with the Bourne shell.

·The Korn shell conforms to the Shell Language Standard which was developed by the IEEE POSIX1003.2 Shell and Utilities Language Committee.

**How Does Your System Know What Shell to**

**Start for You?**



**Notes**

The login program determines whether a person gets access to the system by validating the user name and password against the entries in the /etc/passwd file.

UNIX Shell Programming 1-6

The last field in the password file could be an executableprogram instead of a shell. This is not a valid option for the super user.

If no entry appears in the last field in/etc/passwd, the default/bin/sh shell is executed.



**Shell Startup**

|  |  |
| --- | --- |
| What you see or type | What is happening |
| $ | Your shell indicates its presence |
| 1s Return | You type a command followed by a RETURN |
| Shell searching for the 1s command (You  can't see) | Your shell analyzes what you typed and tries to carry out your request |
| Output of 1s  dev etc tmp unix usr | Your shell sends a request to the  Kernel to execute your command |
| $ | Your shell waits for the program to finish and then indicates its readiness to accept another command. |

**Notes**

The Kernel must copy the requested program into memory before it can begin execution.

UNIX Shell Programming 1-7

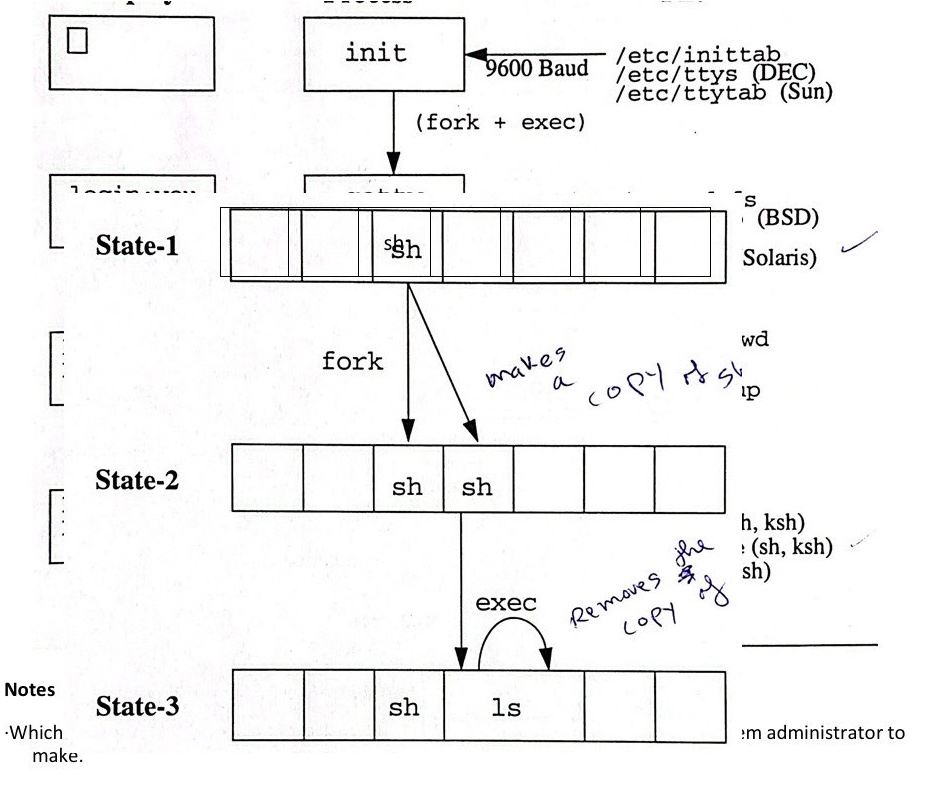
·The program in memory or execution is called a process.

UNIX Shell Programming

**Login Sequence**

Su

Display Process File



**Processes Tables**



UNIX Shell Programming. 1-9

**Notes**

·Information about the processes is kept in process tables.

·A process that creates another process is called a **parent** process.

The process created by the parent process is called the child process.

The parent process createsthe child process by making a copy of itself. This is referenced to as **forking** **a** child process.

The processes are identical except for the process-ID.

The child process inherits a copy of the environment of the parent process.

The exec system call replaces the current process with anew program. The process-ID does not change.

**Shell Questions and Answers**

Q:How does UNIX keep up with processes that are currently executing?

A:The pertinent information of each process is kept in an entry in a table.

Q:How big is this table?

A: It is defined in the Kernel.

Q:What is a fork?

A: The fork call simply makes a duplicate of the parent process in the next available slot.

Q:This seems strange thing to do. Why do it this way?

A:It's fast.

Q: What is the difference between fork and exec?

A:exec overlays the desired process on top of the original process.

UNIX Shell Programming 1-10

©Copyright 1996-2000 GTRC,All Rights Reserved

Q:Why not just start up the secondary process to begin with?

A:Using fork and exec the child process has available to it the environment of the parent process.

Q:What information is included in the environment of a process?

A:Process and group-IDs

Open files

Working directory

File creation mask

Real and effective user and group-IDs

Resource limitations (file size, memory usage)

Signal settings

Named variables

Notes

**Excursion:UNIX System Calls**

childpid=fork()

**Create a new process by copying the currently executing process (including** all **data,environment variables, and open files). Return the child's process-id to the parent;return 0 in the child.**

**execl(program,** **arg0,** arg1, arg2, arg3,..., NULL)

**Overlay the current process with "program”. Preserve the environment variables** **and the open files; set the new program's arguments to arg0,arg1, arg2,....**

wait(&status)

UNIX Shell Programming 1-11

©Copyright 1996-2000 GTRC,All Rights Reserved

**Wait for a child process to terminate. Return** theexit **status** of thechildin **“status”.**

fd = open(filename, mode)

**Open the file named by "filename” for reading/writing/read-write** ifthe **value of** **mode is 0/1/2. Return a "file descriptor" for the open** fileinf.Afile **descriptor is an** **integer starting a 0; open always uses the** smallest **unused descriptor when it** **openw a file.**

close(fd)

**Close thefile on descriptor "fd” so that the descriptor can be reused.**

fd2=dup(fd)

**Return another file descriptor that points to the same open file as pointed** toby“fd”.

**Always use the smallest unused descriptor.**

read(fd, buffer,length)

write(fd,buffer,1ength)

**Read/Write bytes from/to the file open on“fd”.**

**Notes**

**-These calls are the actual C functions called to request** servicefromthekernel.No **additional arguments are required.**

**Starting a Process in C**

**This is the C code required to start a new process and wait for it to complete:**

UNIX Shell Programming 1-12

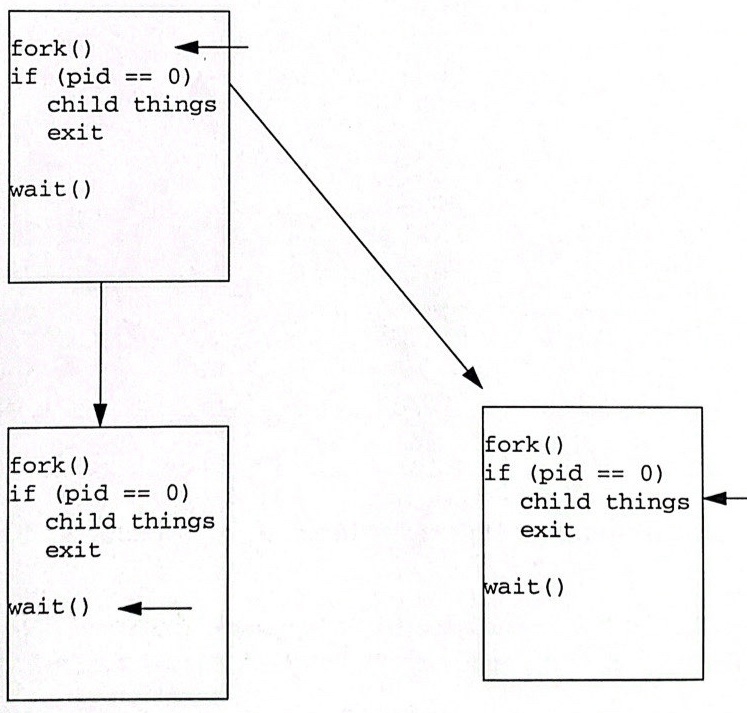
©Copyright 1996-2000 GTRC, All Rights Reserved



/\* do child things \*/

exit(0);

wait(&status) /\* we're the parent, wait for the child \*/



**Running a Program**

This **is the C code required to run the command “Is -1" and wait for it to complete:**

UNIX Shell Programming 1-13

©Copyright 1996-2000 GTRC,All Rights Reserved

C/\* don't need to exit here \*/

wait(&status);



Notes

The variables in the parent's environment **are** **copied** into"ls" during the exec().



Where does"ls"write its output? Good question. Because the child **process** inherits its

parent's open files, “Is” writes to **the** same output file its parent does.

**Who** Opens **the Files?**

/etc/init uses open() to connect file descriptors 0, 1, and 2 to your terminal (or your terminal window application does it when it allocates a pseudo tty device for your window).

getty and then login inherit the open descriptors to your terminal,so they have no trouble finding you.

When login execs your shell,your shell too inherits the descriptors 0,1, and 2 that are open to your terminal.

Your shell wakes up ready to read from your terminal.

Your shell is still free to manipulate those descriptors for itself or for the children it forks.



**Redirecting I/0**

**This is the C code required to run the command “Is -I>myfile":**

UNIX Shell Programming 1-14

if{

1);

/\* don't need to exit here \*/

1

wait(&status);

Notes

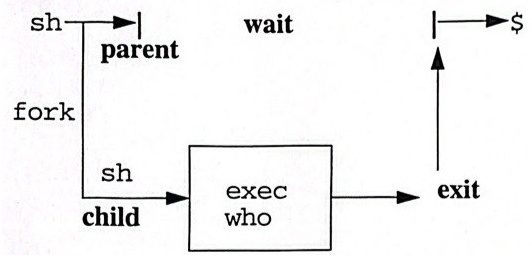
The “standard output" is file descriptor 1, by mutual agreement.By closing it and reopening it pointing to a file in the child process, the output from “Is” will be put into the file.

UNIX Shell Programming 1-16

UNIX trivia: There is a slight omission in the above example if one wants to deal with the case in which “myfile” doesn't exist. The flag values "O CREAT" and"O\_TRUNC"have to be added to the “mode”argument of open() to handle the case properly.

**Shell Process**

$who

Process 1

Process 2

**Process Table**

|  |  |  |
| --- | --- | --- |
| PID | PPID | Command |
| 101 | 98 | sh |
| 121 | 101 | who |

**Notes**

Once the Shell process starts, you will see a system prompt displayed on your screen. You can now issue a command, such as who.

Sequence of Shell steps:

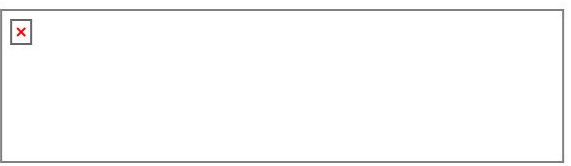
1. Finds who on the disk.

2.Requests Kernel to execute who.

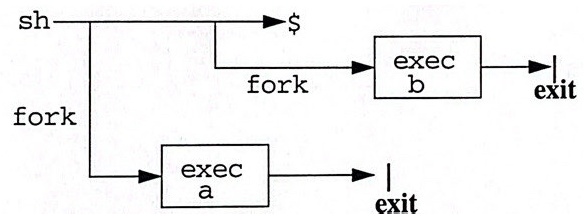
3.Waits for process to finish.

**Foreground/Background Processing**

$a



$a&b&



**Notes**

When the Shell executes a command in the foreground, the parent Shell forks a child Shell which is sacrificed upon an exec of the command. The parent Shell waits for the process to complete.

UNIX Shell Programming 1-17

When the Shell executes a command in the background, the same procedure as above occurs except that the parent Shell does NOT wait for the child process to exit.

Many processes can be run in the background simultaneously.

The ampersand, &, denotes a process is to be run in the background.

To determine what processes are running,use ps.

To terminateforeground or background processes, use ki11along with the PID.

**Additional Shell Responsibilities**

**Command Line:**

$ command [option(s)] [argument(s)]

**·Action items for the Shell:**

**1. Identify command, options and arguments**

**2. Determine if command is built-in**

**3. Perform file name substitution(\*,?,[...])**

**4. Perform I/O redirection**

**5.Identify pipeline characters; connect** whenappropriate

6. Make variable assignments

**Notes**

Before the Shell requests that the Kernel execute the program, the Shell must analyze and interpret information on the command line.

Items 2 and 6 above will be discussed later.

Q:What are the differences between Options and Arguments?

A:Options are usually preceded by a “-".It is a flag or switch to modify the program's execution

Example:1s no option;lists files

1s-1 -1 is the option; gives a long listing

Arguments are usually file name(s) for the command to act upon.

Example cat.profile

.profile is the argument that tells cat which file to act upon.

Usually options come before arguments. Use the verb, direct object, adverb illustration.

Some systems allow you to group options. For example,-a -b can be written as -ab.

**Special Shell Features**

1.Interactive processing

2. Prompt level programming

3.Background processing

4. Input/Output redirection

5.Pipes/Filters

6.Wild card matching:\*,?,[...]

7.Shell scripts

8.Shell variables

**Notes**

· A shell script is a file containing Shell commands and C programs that are executed by the Shell. A Shell script is similar to a BASIC program in that it is interpreted.

The variables are similar to variables in a programming language. Theyare place holders for values. Variables can be assigned by parameter passing, user defined, and Shel defined.

Example:

A file containing the line

grep Fred phonelist