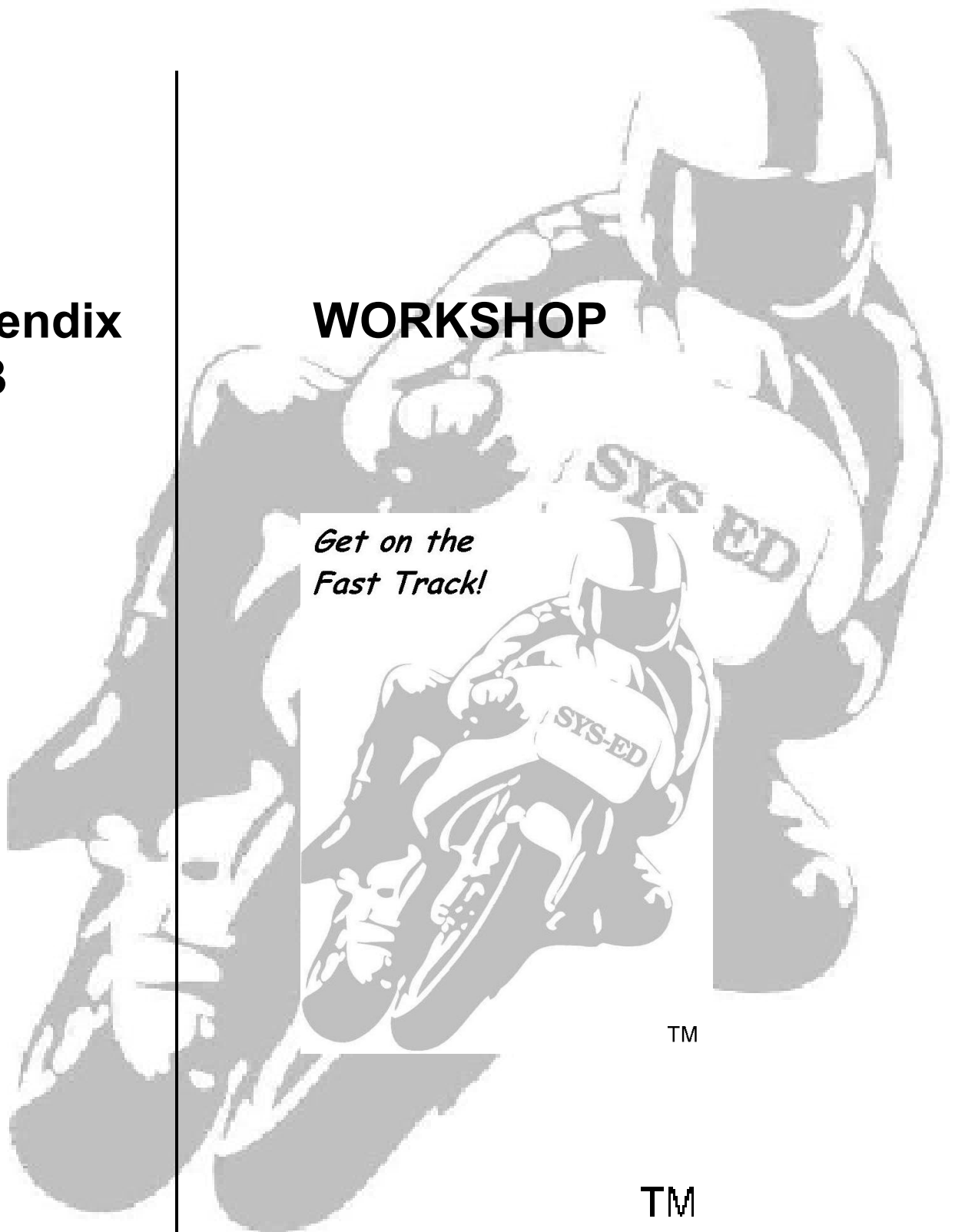


**Appendix
B**

WORKSHOP

*Get on the
Fast Track!*



TM

TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

1 Introduction

Objectives

- Understand the design architecture of the UNIX operating system.
- Become familiar with the leading UNIX-variant operating systems.

There are no machine exercises for this chapter. The instructor will provide walkthroughs and examples.

2 Commands and Utilities

Objectives

- Perform standard operations on files and directories.
- Examine directory and file permissions.
- Execute common UNIX commands and utilities - local and network.
- Use and combine commands for command-line interaction.

Exercises

1. Execute these commands; they require no arguments or options:

date	Show date and time.
whoami	Show your userid.
who	Show who is logged onto the system.
w	Show who is logged onto the system.
pwd	Print the working directory's name.

TM

2. Execute these commands; most will require an argument or option:

ls -a	List all files in current directory.
ls -al	Long list of current directory.
cat .cshrc	Display contents of .cshrc file.
mkdir dir1	Make a directory called dir1
cd dir1	Change directory to dir1.
cd	Change to parent directory.
rmdir dir1	Remove directory dir1.
cp .login new.login	Copy the .login file to new.login.
wc new.login	Count the lines, words, and characters in the new.login file.
wc -l new.login	Count just the lines.
rm new.login	Remove the new.login file.

3. Use multiple commands on a single line; this is done by including the semicolon between commands.

```
cp .login testfile ; cat testfile
    - copy a file and then show its contents

ls -l testfile ; rm testfile ; ls -l testfile
    - list (long) a file, remove it, and then try to
      list it again
```

4. Become familiar with UNIX special characters.

4.1. Use the "*" wildcard character to list all of the files in the present working directory:

```
ls *
```

4.2. Use the "?" wildcard character, to list all files with 3 character names:

```
ls ???
```

4.3. Use square brackets and wildcard characters to list files several different ways:

```
ls [a-c]*
ls [abcde]*
ls [a-z]*
ls [z]*
ls ??[c]*
ls ?[e]*
```

5. Use the right angle bracket ">" and the semi-colon ";" to concatenate three files into a single new file and then display it:

```
cat alpha beta gamma > newfile ; cat newfile
```



3 Files and Directories

Objectives

- Examine and navigate the UNIX file system structure.
- Perform file management and use commands.
- Add files and manage the UNIX environment.

Exercises

1. Create a home directory structure.
2. Use these commands:

mkdir	Make a directory.
rmdir	Remove a directory.
rm	Remove a file.
ls	List files and directories.
pwd	Print working directory.
cp	Copy files and directories.
cd	Change directory.

Examples:

mkdir temp		touch test
		rm test
rmdir temp		
rm temp		pwd
ls		cd
ls -al		
		cd / cd /usr/local/bin
cp /etc/profile		cd temp

3. Independent as to the current location in the file system hierarchy, go to the "home" directory.
 - 3.1. In the home directory, create these subdirectories:
`documents spreadsheets employees`
 - 3.2. Within the "documents" subdirectory, create these subdirectories: **TM**
`letters memos`
 - 3.3. Within the "spreadsheets" directory, create these subdirectories:
`plans budget proposals`

- 3.4. Move into the "budget" directory; what is the current "path"?
- 3.5. Move back to the "home" directory.
- 3.6. Copy the file named "/usr/fred/.exrc" to the home directory.
 - 3.6.1. This file will be used in the "vi" editor exercises.
- 3.7. In the home directory, create a long list of all the files.
 - 3.7.1. Which are "files" and which are "directories"?
 - 3.7.2. How are files differentiated from directories.?

4 File System and Disk Administration

Objectives

- Setting file and directory permissions.
- File system role and functions.
- Understand and use start up files.
- Switch shells.
- Searching for files.
- Compare files.

Exercises

1. Perform these file management exercises.
 - 1.1. What are the current permissions for the home directory?
 - 1.2. What permissions do users in the group have in the current logon home directory?
 - 1.3. What permissions do users not in the group have in the current logon directory?
2. Use the cat command to display the .profile file on the computer screen.
3. Make a backup copy of current logon .profile file; name the backup copy .profile.bak.
4. In the Korn shell, there are two start up files:

"/etc/profile"
".profile", which is located in the home directory.

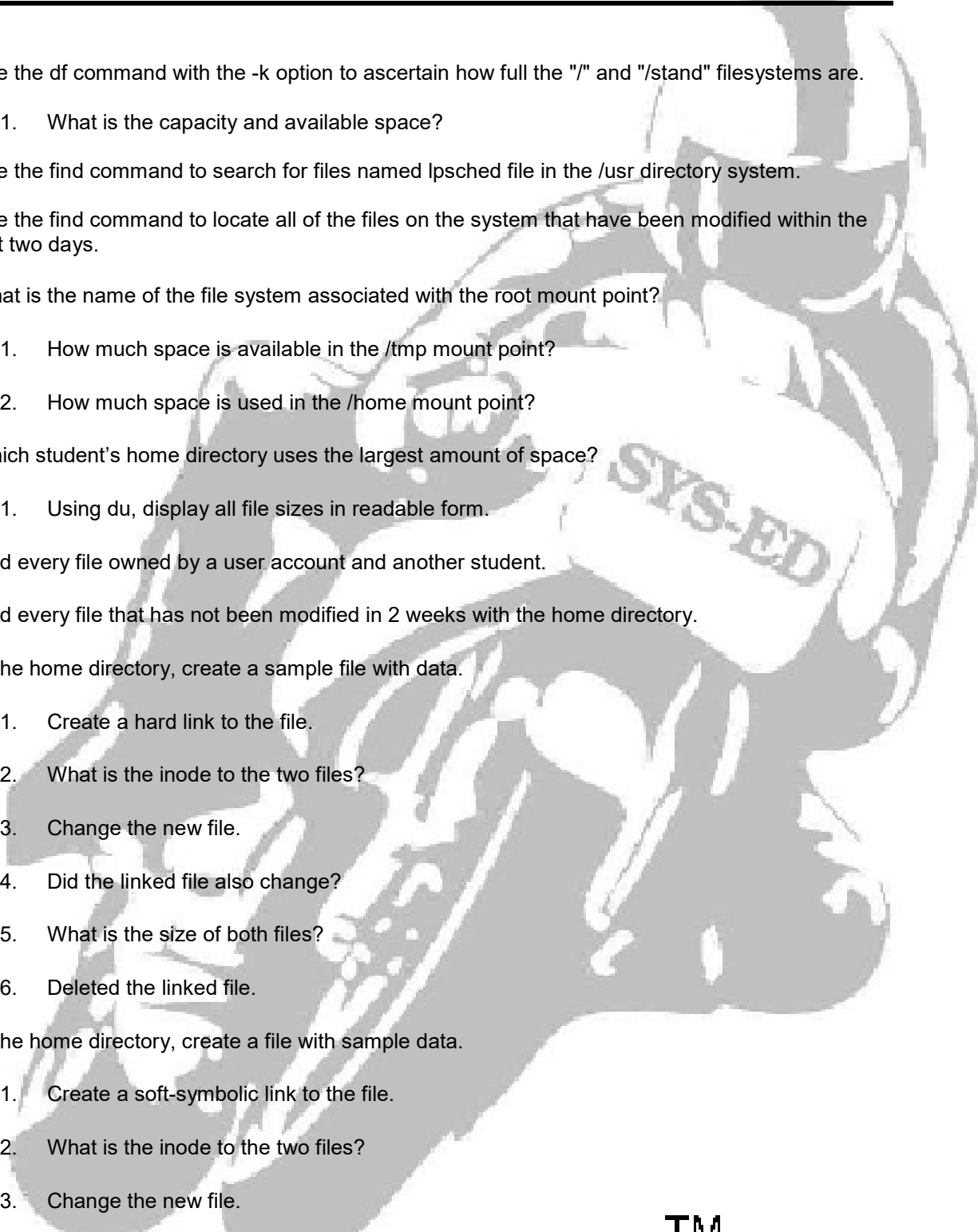
TM

- 5. In the Korn shell, there are two startup files:
 - 5.1. Use the wc command to determine how many lines are contained in each of these files.

File Name	Number of lines
<code>/etc/profile</code> <code>.profile</code>	

- 6. Assume that that the UNIX user account is for a sales manager of a small business in Louisville, Kentucky:
 - 6.1. Create a directory in the home directory called vendors: `/home/stu#/vendors`.
 - 6.2. Create other directories in the home directory called clients, proposals, and misc.
- 7. Move in to the clients directory.
 - 7.1. Create a directory called ky; this stands for Kentucky.
 - 7.2. Create another directory and call it Indiana.
- 8. Move in to the ky directory.
 - 8.1. What is the current working directory/full path?
- 9. The previous sales manager was named "Fred".
 - 9.1. Copy all of the files from `/home/fred/clients/ky` to the ky directory.
 - 9.2. Confirm that all of the files have been copied properly.
- 10. Move into another student's directory and attempt to create a file there with the "touch" command.
 - 10.1. What happens?
 - 10.2. Why?
- 11. Move to different directories on the UNIX file system and use the file command to ascertain the different types of files on the system.
 - 11.1. What types of files are in the `/bin` or `/dev` directories?
- 12. Copy the `/etc/passwd` file to your home directory.
 - 12.1. Confirm that it is there.
 - 12.2. From the home directory delete the passwd file that was created in the misc subdirectory.

TM

- 
13. Use the df command with the -k option to ascertain how full the "/" and "/stand" filesystems are.
 - 13.1. What is the capacity and available space?
 14. Use the find command to search for files named lpsched file in the /usr directory system.
 15. Use the find command to locate all of the files on the system that have been modified within the last two days.
 16. What is the name of the file system associated with the root mount point?
 - 16.1. How much space is available in the /tmp mount point?
 - 16.2. How much space is used in the /home mount point?
 17. Which student's home directory uses the largest amount of space?
 - 17.1. Using du, display all file sizes in readable form.
 18. Find every file owned by a user account and another student.
 19. Find every file that has not been modified in 2 weeks with the home directory.
 20. In the home directory, create a sample file with data.
 - 20.1. Create a hard link to the file.
 - 20.2. What is the inode to the two files?
 - 20.3. Change the new file.
 - 20.4. Did the linked file also change?
 - 20.5. What is the size of both files?
 - 20.6. Deleted the linked file.
 21. In the home directory, create a file with sample data.
 - 21.1. Create a soft-symbolic link to the file.
 - 21.2. What is the inode to the two files?
 - 21.3. Change the new file.
 - 21.4. Did the linked file also change?
 - 21.5. What is the size of both files?
 - 21.6. Delete the linked file.

TM

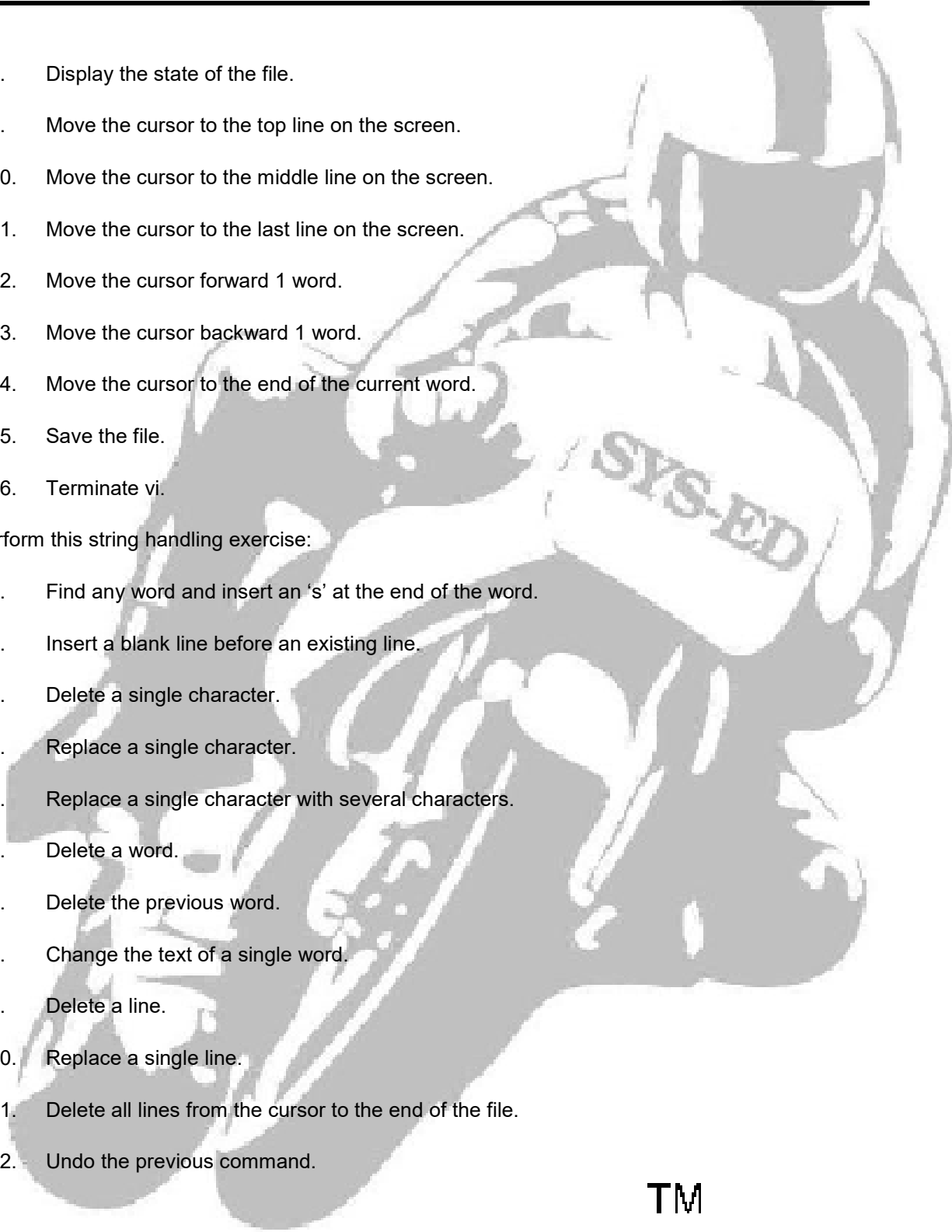
5 vi Text Editor**Objectives**

- Moving around in a file and the screen.
- Operating on lines.
- Moving, rearranging, and duplicating text.
- Searching, goto, and previous content.
- Processing words, sentences, and paragraphs.

Exercises

1. Perform this cursor movement exercise:
 - 1.1. Create a new file using vi.
 - 1.2. Enter a few short paragraphs.
 - 1.3. Use the cursor positioning keys, scroll up, down, left, and right.
 - 1.4. Use the scrolling and paging commands to scroll up and down a page.
 - 1.4.1. Scroll up and down; one line at a time.
 - 1.5. Search for the letter 'a'.
 - 1.5.1. Find the next occurrence of the letter 'a'.
 - 1.6. Search backwards for the letter 'a'.
 - 1.6.1. Find the previous occurrence of the letter 'a'.
 - 1.7. Move the cursor to the 20th line in the file.
 - 1.7.1. Move to the first line.
 - 1.7.2. Move to the last line.
 - 1.7.3. Use the G command to go to a specific line.

TM

- 
- 1.8. Display the state of the file.
 - 1.9. Move the cursor to the top line on the screen.
 - 1.10. Move the cursor to the middle line on the screen.
 - 1.11. Move the cursor to the last line on the screen.
 - 1.12. Move the cursor forward 1 word.
 - 1.13. Move the cursor backward 1 word.
 - 1.14. Move the cursor to the end of the current word.
 - 1.15. Save the file.
 - 1.16. Terminate vi.
 2. Perform this string handling exercise:
 - 2.1. Find any word and insert an 's' at the end of the word.
 - 2.2. Insert a blank line before an existing line.
 - 2.3. Delete a single character.
 - 2.4. Replace a single character.
 - 2.5. Replace a single character with several characters.
 - 2.6. Delete a word.
 - 2.7. Delete the previous word.
 - 2.8. Change the text of a single word.
 - 2.9. Delete a line.
 - 2.10. Replace a single line.
 - 2.11. Delete all lines from the cursor to the end of the file.
 - 2.12. Undo the previous command.

TM

6 Execution Environment

Objectives

- Examine and set environment variables.
- Shell selection and customization.
- Examine and manage process control.

Exercises

1. Processes and job control.
 - 1.1. Display the current processes with the "ps -f" command.
 - 1.1.1. What processes are running?
 - 1.2. What is the PID(s) of the "ksh" process?
 - 1.3. What happen if the PID of the "ksh" process is killed?
 - 1.4. Use the "ps" command to determine the processes being used by user "stu1".
 - 1.5. Use this command for determining how many processes are being run by student accounts:

```
ps -ef | grep stu | wc -l
```
 - 1.6. Use the "-u" option with "ps" to determine what processes are being run by Student #1.
 - 1.7. Attempt to kill the PID of Student #1's "ksh" process.
 - 1.7.1. Record the outcome.



TM

2. Multi-processing walkthrough; type these commands:

Type	Description
<code>cd</code>	Move to your home directory.
<code>vi .profile</code>	Edit your .profile file.
<code>^Z</code>	Suspend your vi job.
<code>vi /etc/profile</code>	Edit the /etc/profile file
<code>^Z</code>	Suspend that vi job.
<code>jobs -l</code>	List the jobs in your job table.
<code>find /usr -name lpsched - print</code>	Search for a file named lpsched.
<code>^Z</code>	Stop the find command.
<code>jobs -l</code>	List the jobs in your job table again.
<code>systat</code>	Run a shell program named systat.
<code>^Z</code>	Suspend that process also.
<code>fg %1</code>	Go back to your first vi process - your .profile file.
<code>:q</code>	Quit vi
<code>jobs -l</code>	List the jobs in your job table again.
<code>fg %2</code>	Bring your second vi editing session back to the foreground.
<code>:q</code>	Quit this vi session also.
<code>jobs -l</code>	List the jobs again.
<code>fg</code>	Bring the most recent job to the foreground.
<code>q</code>	Quit the systat program.
<code>fg</code>	Bring the "find" process back to the active foreground.
<code>^i</code>	Interrupt the find searching process.

TM

7 Shell Programming

Objectives

- Examine and use shell variables.
- Use shell arguments and quotes.
- Use shell standard input and output.
- Use standard device redirection operators.
- Use command substitution.
- Use shell positional parameters.

Exercises

1. Write a shell script which renames all .txt files as .text files.
2. Write a shell script called pidof which takes a name as parameter and returns the PID(s) of processes with that name.
3. Shell scripts also can include functions.

Functions are declared as:

```
function funcname() {  
    statements  
}
```

and invoked as funcname param1 param2...

- 3.1. The parameters passed to the function are accessible inside the function through the variables \$1, \$2, etc.
4. Add a usage() function to the pidof script which prints usage instructions.
5. Call usage() if the wrong number of parameters is passed to the script.

TM

8 Bourne Shell Programming

Objectives

- Use variables.
- Use and set parameters.
- Use the test command and apply testing.
- Use flow of control and looping.
- Apply repeated action commands.
- Use Bourne Shell functions.

Exercises

1. Write a script called greetme that:
 - 1.1. Greets the user.
 - 1.2. Prints the date and time.
 - 1.3. Prints the name of your machine.
2. Write a script called rename that will take two arguments: the name of the original file and the new name for the file.
 - 2.1. If two arguments have not been entered, print an error message and exit.
 - 2.2. Otherwise, rename the file and print a message stating what has been done.
3. Modify the previous program to check for the existence of the original file before performing the rename.
 - 3.1. If it exists, rename it.
 - 3.2. Otherwise print an appropriate error message.
4. Write a script called userInfo that:
 - 4.1. Asks the user's full name: first, middle, and last.
 - 4.2. Greets the user by name and asks for the year of birth.
 - 4.3. Calculate age and output it to the screen with an appropriate message.



TM

5. Write a script called checking that will:
 - 5.1. Take as a command-line argument the user's name.
 - 5.2. Test whether the argument was provided and exited with an appropriate message.
 - 5.3. If the argument was not provided with an appropriate message; check whether the user is in `/etc/passwd`.
 - 5.4. If the argument was not provided with an appropriate message, print a message stating that the user was found or that there was "no such user on our system."

9 Korn Shell Programming

Objectives

- Use Korn shell environment - variables and system variables.
- Conditional expressions.
- Redirecting input and output.
- Design and develop ksh programs.
- Apply ksh flow of control.

Exercises

1. Change the prompt - PS1 to display the current working directory.
2. Change the second prompt to read "continue, please: ".
3. What is the value of the TERM and HOME environment variables?
4. Determine the value of your PATH variable and write it down.
 - 4.1. Modify the PATH variable to include only the `/usr/local` directory.
 - 4.2. Use the `date` command again - what happens?
 - 4.3. Use the `ls` command.
 - 4.4. Use the `pwd` and the `cd` commands.
5. Log out and back in, then use the `whence` command to determine what has transpired.

TM

6. Set the noclobber variable by enabling it.
 - 6.1. Create a file named file_list by redirecting the standard output from an ls command.
 - 6.2. Attempt to overwrite this file by performing the exact same command again.
 - 6.2.1. What happens?

7. Create an alias named dir that you can use that represents this pipeline command:

```
ls -aF | more
```

8. Create an alias named which stands for pwd.
9. Create an alias named del that stands for "rm -i".
10. What is the value of your HISTSIZE variable?
11. Enter these commands:

```
cd
ls

ls -al
ps -ef
who
```

12. Use the fc -l command to list your previous commands.
13. Use the r command to repeat your most recent ls command; refer to the ls command by number.
14. Use the r command to repeat the most recent ps command; reference the ps command by name instead of number.



TM

10 sed**Objectives**

- Perform text processing with sed.
- Use line-oriented patterns in sed.
- Place sed programs into files.
- Apply regular expression substitution.

Exercises

1. Answer these questions.
 - 1.1. What will the command `sed '' filename` do?
 - 1.2. Where does sed normally place its output?
 - 1.3. What will the command `sed '2,4 s/e/#/' filename` do?
 - 1.4. What will the command `sed '2,4 s/e/#/g' filename` do?
 - 1.5. What will the command `sed '2,end s/e/#/' filename` do?
 - 1.6. What does the metacharacter `.o` match? `Cc`
 - 1.7. What does the metacharacter `$` do?
 - 1.8. What does the pattern `^.o` match?
 - 1.9. What is the difference between:
`sed '/^$/d' filename` and
`sed '/^ *$/d' filename` do?
 - 1.10. How does the option `-n` change a sed instruction?
 - 1.11. Enter a sed command that will place 10 spaces at the beginning of each line of file1.
 - 1.12. What does the command `sed '5q' file1` do?
 - 1.13. What does the command `sed '/west/s/5/8' file1` do?
 - 1.14. What does the sed command `sed -e '1,3d' -e 's/Hemenway/Jones/' datafile` do?
 - 1.15. What does the sed command `sed -n '/west/,/east/p' datafile` do?

TM

11 awk Programming

Objectives


- Using awk for searching records.
- Use input lines to awk.
- Utilize the command-line in awk.
- Apply pattern selection.
- Use operators and variables.

Exercises

1. Answer these questions.
 - 1.1. How are escape characters used in awk statements?
 - 1.2. What is the purpose of format specifiers in printf statements?
 - 1.3. What are some differences between print and printf in awk?
 - 1.4. When multiple actions are entered in a awk command, how is this different from entering a single line command versus a multi-line script?
 - 1.5. Explain how the tilde (~) command works in awk?
 - 1.6. What is the role of a pattern versus an action in awk?
 - 1.7. What do the regular expression metacharacters perform in awk?
 - 1.8. Discuss the -F option in awk as it relates to multiple field separators?
 - 1.9. What does the variable \$0 contain in awk?
 - 1.10. What does the command `awk '{print $0}' datafile` do?



TM

- 
2. Use the file provided by the instructor and perform these exercises:
 - 2.1. Enter the awk command that will print all the phone numbers.
 - 2.2. Enter the awk command that will print Dan's phone number.
 - 2.3. Enter the awk command that will print Susan's name and phone number.
 - 2.4. Enter the awk command that will print all last names beginning with D.
 - 2.5. Enter the awk command that will print all first names beginning with C or E.
 - 2.6. Enter the awk command that will print all first names containing only four characters.
 - 2.7. Enter the awk command that will print the first names of all people in the 916 area code.
 - 2.8. Enter the awk command that will print Mike's campaign contributions preceded by a dollar sign - \$.
 - 2.9. Enter the awk command that will print last names followed by a comma and first name. It will be easier than with sed.
 - 2.10. Write an awk script called facts which:
 - 2.10.1. Prints full names and phone numbers for the Savages.
 - 2.10.2. Prints Chet's contributions.
 - 2.10.3. Prints all those who contributed \$250 the first month.

TM