



AIX 6 Basics

(Course Code AU13)

Student Exercises

ERC 10.0

IBM Certified Course Material

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX®	AIX 5L™	Common User Access®
MVS™	OS/2®	pSeries®
System p™	System p5™	400®

PS/2® is a trademark or registered trademark of Lenovo in the United States, other countries, or both.

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX® is a registered trademark of The Open Group in the United States and other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

February 2008 Edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright International Business Machines Corporation 1995, 2008. All rights reserved.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	v
Exercises Description	vii
Exercise 1. Using the System	1-1
Exercise 2. AIX 6.1 Documentation	2-1
Exercise 3. Files and Directories	3-1
Exercise 4. Using Files	4-1
Exercise 5. File Permissions	5-1
Exercise 6. vi Editor	6-1
Exercise 7. Shell Basics	7-1
Exercise 8. Using Shell Variables	8-1
Exercise 9. Controlling Processes	9-1
Exercise 10. Customizing the User Environment	10-1
Exercise 11. AIX Utilities (1)	11-1
Exercise 12. AIX Utilities (2)	12-1
Exercise 13. AIX Utilities (3)	13-1
Exercise 14. AIX Utilities (4)	14-1
Exercise 15. Additional Shell Features	15-1
Exercise 16. Using AIXwindows	16-1
Exercise 17. Using the Common Desktop Environment (CDE)	17-1
Appendix A. Customizing AIXwindows (1)	A-1
Appendix B. Customizing AIXwindows (2)	B-1
Appendix C. Customizing CDE	C-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX®	AIX 5L™	Common User Access®
MVS™	OS/2®	pSeries®
System p™	System p5™	400®

PS/2® is a trademark or registered trademark of Lenovo in the United States, other countries, or both.

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX® is a registered trademark of The Open Group in the United States and other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Exercises Description

None of the exercises, EXCEPT Exercises 4 and 5 are dependent on the preceding exercise being successfully completed. It is assumed, however, that you understand the commands and concepts from each exercise as these commands and concepts are carried over to the follow-on exercises.

Each exercise in this course is divided into sections as described below. Select the section that best fits your method of performing exercises. You may select to use a combination of these sections as appropriate.

Exercise Instructions - This section contains what it is you are to accomplish. There are no definitive details on how to perform the tasks. You are given the opportunity to work through the exercise given what you learned in the unit presentation, utilizing the unit Student Notebook, your past experience, and maybe a little intuition.

Exercise Instructions with Hints - This section is an exact duplicate of the **Exercise Instructions** section except that in addition, specific details and hints are provided to help step you through the exercise. Using the **Exercise Instructions** section along with the **Exercise Instructions with Hints** section can make for a rewarding combination providing you with no hints when you do not want them and hints when you need them. When there is more than one way to do a command, we show you both ways with an **-OR-** between possible solutions.

Optional Exercises - This section provides additional practice on a particular topic. Specific details and hints are provided to help step you through the **Optional Exercises**, if needed. Not all exercises include **Optional Exercises**.

Solutions - This section provides at least one solution to questions strategically placed in some exercises. Where applicable the solutions have been provided at the end of the **Exercise Instructions with Hints** section. Note: These are NOT the solutions to the exercises as those are provided in the **Exercise Instructions with Hints**.

Text highlighting

The following text highlighting conventions are used throughout this book:

Bold	Identifies file names, file paths, directories, user names, and principals.
<i>Italics</i>	Identifies links to Web sites, publication titles, and is used where the word or phrase is meant to stand out from the surrounding text.
Monospace	Identifies attributes, variables, file listings, SMIT menus, code examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, and messages from the system.
Monospace bold	Identifies commands, daemons, menu paths, and what the user would enter in examples of commands and SMIT menus.
<text>	The text between the < and > symbols identifies information the user must supply. The text may be normal highlighting, bold or monospace, or monospace bold depending on the context.

Exercise 1. Using the System

What This Exercise Is About

The purpose of this exercise is to become familiar with AIX command syntax and basic commands.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- Log in to an AIX system and change passwords
- Execute basic commands
- Use the **wall** and **write** commands to communicate with other users
- Use keyboard control keys to control command line output

Introduction

When executing commands on the command line, use the **Enter** key on the graphics keyboard not the **Ctrl/Act** key. If using an ASCII keyboard use the **Return** key not the **Send** key. Use of the **Ctrl/Act** or **Send** keys can cause unpredictable results. When correcting a typographical error on the command line, use the **Backspace** key not the **arrow** keys.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.

Logging In / Changing Passwords

- ___ 1. Log in to the system with the user name and password provided by your instructor. It should be a user name such as **teamxx** where **xx** is a double digit number like **01**, **02** and so forth.

The very first time you log in with your new user name, you will be prompted to change your password. Keep the password the same as your login name. The passwords you supply on the command line will not be displayed.

- ___ 2. Verify that the password has been set by logging out and back in.

Basic Commands

- ___ 3. Display the system's date.
- ___ 4. Display the whole calendar for the year 2007.
- ___ 5. Display the month of September for the year 1752. Notice anything peculiar about September? _____
- ___ 6. Display the month of January for the years 1999 and 99. Are 1999 and 99 the same? _____
- ___ 7. There are two commands that will display information about all users currently on the local system. Display who is currently logged in on your system. Check to see when they logged in.
- ___ 8. Display just your login name.
- ___ 9. Use **banner** to display `Out to Lunch`.
- ___ 10. Use the **echo** command to write the character string `Out to Lunch` to your display.
- ___ 11. Use the **clear** command to clear your screen.

Send and Receive Mail

- ___ 12. Send a note to yourself using the **mail** command. Provide a subject but ignore the carbon copy prompt.

- ___ 13. Start the **mail** process and list the message in your mailbox. Read your message, save it, and quit the **mail** program. To list a brief summary of **mail** subcommands, type **?** at the **mail** prompt.
- ___ 14. Access your mail and delete the message you saved in your personal mailbox. Exit the **mail** program. If there is more than one person logged in on your system, practice sending mail to each other.

Communicating with Other Users

- ___ 15. Send a note to all users on the system indicating that you have almost completed this exercise.
- ___ 16. Pair up with someone on your system to coordinate this exercise. Open a line of communication to send a message to your partner, **teamyy**. Let teamyy know that you are waiting for a response. **teamyy** should then reply and let you know that they have nothing else to say. End of conversation.

Keyboard Tips

To get some practice temporarily stopping, starting, and terminating the scrolling of command output, use the **banner** command to banner the letters of the alphabet in order to generate multiple lines of output.

- ___ 17. Using **banner**, display the alphabet separating each character with a space. As output is scrolling to your display, temporarily stop the output. Resume the scrolling.
- ___ 18. Repeat the **banner** command used in the previous step, typing only the first five letters of the alphabet, but DO NOT press **Enter**. Erase your input using **<Ctrl-u>**. Now have the **banner** command display the phrase **End of Exercise**. This time if you make a typing mistake while keying this command, use the **Backspace** key to correct the command line.
- ___ 19. Log off the system.

END OF EXERCISE

Exercise 2. AIX 6.1 Documentation

What This Exercise Is About

The purpose of this exercise is to give the students the opportunity to explore and experiment with the **man** command and with the AIX 6.1 online documentation.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- Execute the **man** command
- Start a Web browser to access the online documentation

Introduction

In this exercise, you will first use the **man** command from the command line. This part of the exercise can be performed in either graphics mode or ASCII mode.

In the second part of the exercise, you will use a Web browser to access AIX 6.1 online documentation.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.

man Pages

- ___ 1. Log in to the system with the user name and password provided by your instructor.
- ___ 2. Bring up the **man** pages for the **man** command. Read the text that follows to obtain a better understanding of the functionality of the **man** command.

Remember to use the space bar to go forward one screen and the return key to go forward one line. Press the **b** key to go back one screen. When you have read enough, exit **man** using the **q** key or **<Ctrl-c>**.

- ___ 3. Using the **man** command, search on the keyword **calendar**. From the list produced, find the command that displays a calendar.
- ___ 4. Having found the **cal** command from the previous step, use **man** without any options to obtain the correct syntax of the command.

AIX Information Center

- ___ 5. Start up a Web browser and access the online documentation. Your instructor will tell you whether to use the Internet site or a local AIX system configured as a documentation server.
The URL for the Internet site is:
<http://publib.boulder.ibm.com/infocenter/pseries/v6r1/index.jsp> Click the **AIX Information** link in the left frame.
- ___ 6. In the right frame of your Web browser, click the link entitled *Operating System and Device Management*.
- ___ 7. Select one or two of the topics displayed in the right frame.
- ___ 8. Now, suppose you do not know what document to look in for the information you require. Use the search function in the Information Center to find information on the **wc** command.

»

- ___ 9. Use the **Search Scope** function to narrow your search. Change the search scope for the previous search to only include the Commands Reference.

»

- ___ 10. Use any extra time you have to explore other documents available in the AIX Information Center.
- ___ 11. Exit your Web browser.

>>

END OF EXERCISE

Exercise 3. Files and Directories

What This Exercise Is About

This exercise provides the students with the opportunity to begin working with directories and the files they contain.

What You Should Be Able to Do

After completing this exercise, you should be able to:

- Display the name of the current directory
- Change directories
- Use various options of the `ls` command to display information about files and directories
- Create and remove directories
- Create zero-length files

Introduction

In this exercise, you will be using AIX commands to work with directories and files.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.
- ___ 1. If you are not already logged in, log in to the system
 - ___ 2. Using the **pwd** command, verify that you are in your home directory, **/home/teamxx**, the directory where you are placed when you first log in.
 - ___ 3. Change your current directory to the root directory (**/**).
 - ___ 4. Verify that you are in the root directory and then execute both a simple and a long listing of the files in that directory.
 - ___ 5. Issue the **ls** command with the **-a** and the **-R** options. What is the effect of each option? _____ (**Note:** The **ls -R** will provide extensive output. Once you have seen enough, enter the key sequence **<Ctrl-C>** to end the command.)
 - ___ 6. Return to your home directory (**/home/teamxx**) and list its contents including hidden files.
 - ___ 7. Create a directory in your home directory called **mydir**. Then, issue commands to view a long listing of both your **/home/teamxx/mydir** and **/home/teamxx** directories. What are the sizes of each directory? _____
 - ___ 8. Change to the **/home/teamxx/mydir** directory. Use the **touch** command to create two zero-length files called **myfile1** and **myfile2** in your **mydir** directory.
 - ___ 9. Issue the command to view a long listing of the contents of your **mydir** directory. What are the sizes of **myfile1** and **myfile2**? _____ View the long listing again, this time also displaying the i-node numbers of the files. What are the i-node numbers for the files? _____
 - ___ 10. Change back to your home directory and issue the **ls -R** command to view your directory tree.
 - ___ 11. Use the **lsstat** command to view i-node information on your **mydir** directory. Why may the "Last Accessed" date be more current than the other two dates?
 - ___ 12. Use the **rmdir** command to remove the **mydir** directory. Does it work?
_____ You will note that the **rmdir** command cannot remove a non-empty directory. To do that, you will need to issue a command that we will learn in the next unit, **rm -r**.

END OF EXERCISE

Exercise 4. Using Files

What This Exercise Is About

In this exercise, students use a number of AIX commands to manipulate files.

What You Should Be Able to Do

After completing this exercise, a student should be able to:

- Copy, move, rename, link, and remove files
- Display the contents of a file
- Print a file

Introduction

In this exercise you will be using AIX commands to manipulate ordinary files and directories using the commands discussed in lecture.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.

Checking Your Environment

- ___ 1. If not already logged in, log in to the system.
- ___ 2. Using **pwd**, verify that you are in your home directory, **/home/teamxx**, the directory where you are placed when you first log in.
- ___ 3. List the contents of your home directory (**/home/teamxx**), including hidden files.

Working with Files

- ___ 4. Look at the contents of the **/etc/hosts** and **/etc/profile** files. Use the commands **cat**, **pg**, and **more** to see how each command handles the output.
- ___ 5. Copy the file **/usr/bin/cat** into your current (home) directory.
- ___ 6. Copy the file **/usr/bin/cal** into your current (home) directory.
- ___ 7. List the files in your current directory. You should see the two you just copied.

Creating and Manipulating Directories

- ___ 8. Create a subdirectory in your home directory called **myscripts**.
- ___ 9. Move and rename the two files that you just copied to your home directory (**cat** and **cal**) into your new subdirectory. Name the new files **mycat** and **mycal** respectively.
- ___ 10. Make the new subdirectory, **myscripts**, your current directory.
- ___ 11. List the contents of the directory to make sure that the files were copied.
- ___ 12. Use the **mycat** command in your **myscripts** directory to look at the contents of the **.profile** file in your home directory.
- ___ 13. Make your home directory the current directory.
- ___ 14. Create another subdirectory in your home directory called **goodstuff**.
- ___ 15. Copy a file called **/etc/profile** into the new directory, and name the new file **newprofile**.
- ___ 16. Use the **cat** command to look at the file. Hard to read? Try the **pg** command.
- ___ 17. The filename, **newprofile**, is too long to keep typing. Change its name to **np**. List the contents of the **goodstuff** directory to make sure that you have accomplished the task. Use the **cat** command to type out the renamed file.

- ___ 18. This is a good point to check everything out. Starting from your home directory and working downwards, display a hierarchical tree of your files and subdirectories.

Remove a Directory

- ___ 19. Ensure you are in your home directory. Remove the **goodstuff** directory. Could you do it? Why or why not?
- ___ 20. Change to the **goodstuff** directory. Do a listing on the contents of the **goodstuff** directory including any hidden files. Remove the files. Do another listing on the **goodstuff** directory including the hidden files. Notice the **.** and **..** files are still there. The directory is considered “empty” if these are the only two entries left in it. Remove the directory.

END OF EXERCISE

Optional Exercises

- ___ 21. Using the **mkdir** command only once, create a directory under the **myscripts** directory named **sports** that has three directories in it named **tennis**, **basketball**, and **baseball**. Check to be sure the directories were created properly.
- ___ 22. Copy the file **/etc/motd** into the **tennis** directory and create two files in the **basketball** directory. Leave the **baseball** directory empty. Check to be sure the files were created.
- ___ 23. Use the **rm** command to remove the **sports** directory and everything in it.

END OF OPTIONAL EXERCISES

Exercise 5. File Permissions

What This Exercise Is About

The purpose of this exercise is to give the student the opportunity to work with file and directory permissions. A fundamental understanding of basic AIX file ownership and permissions should be a result of performing these exercises.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- Manipulate permissions on ordinary files and directories
- Interpret file and directory permission bits
- Display long listing information for files and directories

Introduction

In this exercise, you will be using AIX commands to manipulate AIX file and directory permissions. Understanding the implications of file permissions and ownership and using the commands to change file permissions is necessary to doing additional exercises in this course.

Tips

Make sure you are aware of what directory you are in while performing the various steps. If you lose track of where you are in the exercise, some instructions will appear not to work. Use `pwd` frequently to check your current directory.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.

Listing Information on Files

- ___ 1. Log in to the system. Change to the **myscripts** directory. Display a long listing of the files in the **myscripts** directory. Notice the owner and permissions for the files that you copied in the previous exercise.

Record the permissions for **mycat**. _____

Record the permissions for **mycal**. _____

- ___ 2. Now, do a long list on the original **cat** and **cal** files in the **/usr/bin** directory and compare the permissions to those in the **myscripts** directory. You own the copies but not the originals.

- ___ 3. Change the modification time of **mycal** and **mycat** in the **myscripts** directory. Check to see that the time actually changed. What is another use for the **touch** command? _____

- ___ 4. Make it so you can reference the **mycal** file in the **myscripts** directory by the name of **home_mycal** in your home directory. Compare the detailed file information for both files.

Is there any difference? _____

What is the link count? _____

- ___ 5. Change the directory to your home directory. Execute **home_mycal**.

What does the output look like?

Now, change permissions on the **home_mycal** file so that you, the owner of the file, have read only permission. Try running the **mycal** command.

Can you do it? _____

Why or why not? _____

- ___ 6. Remove **home_mycal**. Did that remove **myscripts/mycal**? _____

Why or why not?

Working with File Permissions

- ___ 7. Change the directory to the **myscripts** directory. Using symbolic notation of the **chmod** command, remove the read permission on the “other” permission bits from the file **mycat**. Check the new permissions.
- ___ 8. Using octal notation, change the permissions on **mycat** so that the “owner” permission bits are set to read-only permission with no permission for anyone else. Check the new permissions.
- ___ 9. Use the **mycat** command to display the contents of the **.profile** file. Did it work? What happened?

- ___ 10. Make your home directory the current directory. Check to see if you are in your home directory.

Working with Directory Permissions

- ___ 11. Alter the permissions on the **myscripts** directory so that you have read-only access to it.
- ___ 12. Use a long list to check that you have set the permissions correctly.
- ___ 13. Try getting a simple list of the contents of the directory. Try a long list. Did they work? Why or why not?

- ___ 14. Try to execute **mycal**. Did it work? Why or why not?

- ___ 15. Try to remove **mycal**. Did it work? Why or why not?

- ___ 16. Return the permissions of **myscripts** back to its original form of **rxwx-rx-x** and then remove **mycal**.
- ___ 17. As time permits, experiment with other permission combinations. When you are through, make sure to change the permissions back to **rxw** for the owner.

END OF EXERCISE

Exercise 6. vi Editor

What This Exercise Is About

The purpose of this exercise is to give the student the opportunity to create and edit files using the most common UNIX editor, **vi**. A clear understanding of the vi editor is critical to successfully complete the rest of the exercises in this course.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- Create a file
- Save and exit a file and exit without saving
- Manipulate a file using various cursor movement keys
- Add, delete, and make changes to text within a file
- Set options to customize the editing session
- Invoke command line editing

Introduction

The **vi** editor is based on software developed by the University of California at Berkeley, California, Computer Science Division. The **vi** editor, pronounced “vee-eye” (short for visual), features commands to create, change, append, or delete files. The following exercises will familiarize you with some of the major features and functions of **vi**.

For your assistance, there is a **vi** Command Summary in the Appendix of the Student Notebook.

Exercise Instructions

Creating a File

- ___ 1. Ensure that you are in your home directory. Create a file in your home directory named **vitest**.
- ___ 2. When you open a **vi** file, you are automatically placed in command mode. Press the **i** key (insert) to switch to input (text) mode. You can also press the **a** key (append). Use of **i** or **a** simply determines if typing starts before or after the cursor. There is no indication to tell you that you are in input mode.

Switch from input mode to command mode by pressing the **ESC** key. Press **ESC** a second time. Notice that if you press **ESC** twice, you will get a “beep” from the terminal (some ASCII terminals do not beep). The beep indicates that you are in command mode already. Now press **i** again to put you back in input mode. Continue to the next step.

- ___ 3. Input the following text *exactly* as it is presented line-by-line. Then key in the alphabet, one character per line. Following will show **a–d** but continue on through **z**. Adding the alphabet is an easy way to fill a couple of screens of information needed for later use.

```
This is a training session about the usage of the vi editor. We need
some more lines to learn the most common commands of the editor. We are
now in the entry mode and we will switch right after this to the
command mode.
```

```
a
b
c
d
...
z
```

- ___ 4. Return to command mode. Write and quit the file. Notice that as soon as you press the **:** (colon), it appears below the last line of your input area. Once the buffer is empty and the file is closed, you will see a message giving the number of lines and characters in the file.

Cursor Movement Keys

- ___ 5. Open **vitest** using **vi**. Notice the bottom line of the file indicates the name of the file and number of characters.
- ___ 6. Using both the arrow keys and the **h**, **j**, **k**, **l** keys, practice moving the cursor down one line, up one line, right a couple characters, and back a couple characters.

-
- ___ 7. You may not want to cursor one character or one line at a time throughout an entire file. Practice using cursor movement keys to work around by page or by line. Using the cursor movement keys from the previous step, position your cursor at the first line of the file. While in command mode, do the following:
- i. Move forward one page.
 - ii. Move back one page.
 - iii. Move cursor to last line in the file.
 - iv. Move cursor to first line in the file.
 - v. Move cursor to line 4 of the file.
 - vi. Move cursor to end of line.
 - vii. Move cursor to beginning of line.
- ___ 8. Move your cursor to the top of the file. Search for the word `entry`. Your cursor should be on the `e`. Switch to input mode and add the word "`text`". Do not forget the space after the word.
- ___ 9. Move the cursor to the space after the word `mode` on the same line. Insert a comma. Remember, you are still in input mode.
- ___ 10. Enter command mode. Position the cursor anywhere on the line beginning with "`some more lines`". Insert a blank line to form two paragraphs.
- ___ 11. Opening a blank line as in the previous step, automatically puts you in input mode; therefore, return to command mode. Now save the changes you have made so far, but DO NOT exit the editor.
- ___ 12. While still in command mode, remove the alphabetic characters `c`, `e`, `g` but leave the blank lines in their place; in other words, do not delete the entire line, just the character. Then go back and remove the blank lines. This will give you practice using two of the delete functions.
- ___ 13. Now replace the alphabetic character `h` with a `z`.
- ___ 14. You just decided you really do not want to save the changes to the alphabetic characters. Quit the editing session without saving the changes made since the last save.
- ___ 15. Edit **vitest** one more time. First, copy the first paragraph (including the blank line) one line at a time to the end of the file. When that is complete, copy the second paragraph all at once to the end of the file.
- ___ 16. You just decided that the lines you just added to the end of file do not look right. Delete them all with one command.
- ___ 17. Now, before you do anything else with this file, you decide you need to imbed the current date and time as the first line of the file. Do this without leaving the `vi` editor.

Using Set to Customize the Editing Session

- ___ 18. Options can be set temporarily in an editing session using the **set** command. Go back to the top of your file. Ensure you are in command mode and set the following commands:
 - a. Set automatic word wrap 15 spaces before the right margin.
 - b. Display the INPUT MODE message when in input mode.
 - c. Turn line numbering on
- ___ 19. Test each of the options set in the previous instruction.
- ___ 20. Write the file and quit the editor.

Command Line Editing

- ___ 21. Now that you are familiar with **vi** modes and commands, practice command line editing. To set up your session to use command line editing, use the **set -o vi** command.
- ___ 22. Now you can recall previously executed commands, edit them, and resubmit them. Let's build a command history to work with. List (simple, not long) the contents of the directory **/usr**. Display the contents of the file **/etc/filesystems**. Echo **hello**.
- ___ 23. Suppose you want to edit one of the commands you just executed. Press the **ESC** key to get to **vi** command mode. Try pressing the **k** key several times to go up the list of commands. Try **j** to go down. This recall of commands is essentially looking through a buffer of commands that you previously executed. The commands are actually stored in your **.sh_history** file in your home directory.
- ___ 24. Retrieve the **ls** command. Use the **l** key to move your cursor to the **/** in **/usr**. (Note: the arrow keys tend to wipe your line out. You have to use the **l** key for right and **h** for left.) Use the **i** key to insert text and change this command to be a long list. Execute it.
- ___ 25. Recall the **cat** command. This time list the contents of the **/etc/passwd** file.
- ___ 26. Recall the **cat** command. Go to the end of the line (remember **\$**). Add to the end of the command to pipe the output to **wc** to count just the lines.

END OF EXERCISE

Exercise 7. Shell Basics

What This Exercise Is About

This exercise will familiarize the students with basic shell operations.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- Use wildcards for file name expansion
- Redirect standard in, standard out, and standard error
- Use pipes to provide the output of one process as input to another process
- Perform command grouping and line continuation

Introduction

Understanding the use and manipulation of the shell is considered a foundation for understanding AIX user interfaces. You will use commands to experiment with the shell features discussed in the “Shell Basics” lecture.

Exercise Instructions

Wildcards

- ___ 1. Type `cd` to get back to your home directory. (Your home directory is the one you use when you log in.)
- ___ 2. Execute a simple `ls` to list the non-hidden files in your home directory. Now use the `ls` command with a wildcard character to list these files. What is the difference in output of these two commands?

Why?

- ___ 3. Change to the `/usr/bin` directory. List just those files starting with the letter `a`.
- ___ 4. List all two character file names.
- ___ 5. List all file names starting with the letters `a`, `b`, `c`, or `d`.
- ___ 6. List all files except those beginning with `c` through `t`. This will be a long list. You might want to pipe the output to `pg` or `more`. Did you get any file names that you did not expect? _____. If so, do you know why?
- ___ 7. Return to your home directory.

Redirection

- ___ 8. Using the `cat` command and redirection, create a file called `junk` containing a few lines of text. Use `<Ctrl-d>` at the beginning of a new line when you have finished entering text and want to return the shell `$` prompt. List the file contents to verify your update.
- ___ 9. Append more lines of text to the file you have created using the `cat` command and redirection. List the file contents to verify your update.
- ___ 10. Mail the file `junk` to yourself. Wait a minute and open your mail, delete it, and quit the program.

Pipes, Tees, and Filters

- ___ 11. Using the `ls` command, list the files in your current directory. Make a note of the number of files:
- ___ 12. List the files in your current directory, but this time redirect the output to the file `temp`.
- ___ 13. Use the appropriate command to count the number of words in the `temp` file. Is this the same count as in instruction 11? _____. If not, why not?

Display the contents of **temp**. Remove the file.

- ___ 14. This time use a pipe to count the number of files in your current directory. Was the result what you expected this time? _____ Is it the same as in instruction 11?

- ___ 15. Use the command you created in instruction 14, but this time insert a **tee** in the middle trapping the result of the list in a file called **junk2**. Did you get the number displayed on the screen? _____

Check the contents of **junk2** to make sure that it contains what you expected.

»

- ___ 16. List in reverse order the contents of your current directory. Send the results of the reverse listing to a file named **junk3**, and to a program to count the number of words in the reverse listing. Append the final count to **junk3**. Remember to use the append version of redirection. In this particular case, you may get unexpected results if you do not. It might not be a straight overwrite because the file is being used twice in the same command. Experiment if you are curious.

- ___ 17. There is a special file in the **/dev** directory that represents your terminal. Display the file name associated with your terminal. Output will be something like `ttty0`, `lft0`, or `pts/x`. Repeat the command from instruction 16 with two exceptions:

- 1) Rather than using **junk3**, tee the output to the special file that represents your terminal (**/dev/<your_terminal_name>**).
- 2) Do not append the results of the `wc` command to **junk3**. Have the count display to your terminal.

Command Grouping and Line Continuation

- ___ 18. On the same command line, display the date, who is logged in, the name of your current directory, and the names of the files in your current directory. Do these commands have any relationship to each other?
- ___ 19. The primary purpose of this exercise instruction is to use line continuation with a command that is too long to fit on one command line. The secondary purpose is to test what you have learned so far by letting you create an incredibly long command string.

You can choose to break the line anywhere you feel comfortable, but do not type past the right edge of the screen. When completed, test your output by displaying the contents of the files that were created. This should be one long command connected by pipes and redirection.

- 1) Do a long listing of the files in your home directory including hidden files.

- 2) Capture the output to a file named **reverse.listing** and send the same output to a program that will count only the number of words.
- 3) Capture the number of words and place the number in 4 files named **file1** through **file4**.
- 4) Finally, send the output to a program to count the number of lines captured in the previous instruction and redirect that number to a file named **file5**.

END OF EXERCISE

Exercise 8. Using Shell Variables

What This Exercise Is About

The student will define and utilize variable and command substitution to set the shell environment and utilize quoting to override the shell interpretation of metacharacters.

What You Should Be Able to Do

After completing this exercise, students should be able to:

- List shell built-in variables
- Set up variable substitution to define or alter the environment
- Use command substitution to set variables equal to the output of a command
- Use the three methods of quoting to allow metacharacters to be used literally instead of interpreted

Introduction

This exercise contains three sections: variable substitution, command substitution, and quoting. Knowledge of the first two sections, variable and command substitution, is required to perform the third section, quoting.

Caution: Throughout this exercise, the single quotes and the back quotes look very similar. The single quotes look like this `'`, and the back quotes like this ```. The back quote may look different on the keyboard than it does as printed in this exercise.

Exercise Instructions

Variable Substitution

- ___ 1. Display the shell built-in variables.
- ___ 2. Set a variable named `lunch` to `pizza` and a variable named `dinner` to `ham`. Display the value of the variables using `echo`. Locate them in the list of variables.
- ___ 3. Using the variables you just defined, display the message, `Lunch today is pizza` and `dinner is ham`.
- ___ 4. Using the variables you just defined, display the message, `Lunch today is hamburgers`.
- ___ 5. Remove the value of both variables. Check to be sure they are no longer included in your list of variables.
- ___ 6. Display the value of your primary and secondary prompt strings.
- ___ 7. Change the primary prompt string to `"You Rang?"`. (Single quotes will also work) Why is it necessary to use the quotes with `"You Rang?"`?
- ___ 8. Change your secondary prompt string to `"What Else?"`. Test it with the `ls` command using line continuation. End the command. Reset both prompt strings back to their original values. Why are quotes needed around the `>` when resetting the `PS2` variable? _____
- ___ 9. Check the value of the variable related to your home directory. Reset that variable to change your home directory to `/bin`. Use the `cd` and `pwd` commands to test the effects of this change.
- ___ 10. Log out and log back in. What is your home directory? _____ Why?

Note: If you are working in an `aixterm` session, after keying `exit`, press the right mouse button and select `New Window` to get back to an `aixterm` session.

Command Substitution

- ___ 11. Display your list of variables. Reissue the command but send the output to the `wc` command to get the number of variables that are currently set.
- ___ 12. Using command substitution, `echo` the following:
`There are # variables currently set`
`where # is the number of variables.`
- ___ 13. Each user ID configured on the system is represented by one line in the `/etc/passwd` file. Applying your knowledge of command substitution, echo a message that displays:
`There are # users created on the system`
`where # is the number of line entries in /etc/passwd.`

Quoting

- ___ 14. Using all three methods of quoting, **banner** the literal symbol *. Why do all three work?
- ___ 15. Ensure you are in your home directory. Create a directory in your home directory named **quoting**.
- ___ 16. Change to the **quoting** directory. Create a zero-length file in the **quoting** directory named **filea**. Create a variable named `n` set to the value of `hello`. Test what you have done by displaying the contents of **quoting** and the value of `n`.
- ___ 17. From the **quoting** directory, execute the following five commands. Record the output. Check the *Solutions* section for the expected output.

i. `$ echo '* $n `ls` $(ls)'`

ii. `$ echo "* $n `ls` $(ls)"`

iii. `$ echo * \ $n \ `ls\` \ $\ (ls\)`

iv. `$ echo * $n `ls` $(ls)`

v. `$ echo * $n ls`

END OF EXERCISE

Exercise 9. Controlling Processes

What This Exercise Is About

This exercise familiarizes the student with process manipulation and process control.

What You Should Be Able to Do

After completing this exercise students should be able to:

- Monitor processes by using the `ps` or `jobs` command
- Control processes by using the `kill` or `jobs` command
- Display current process ID

Introduction

In this exercise you will use commands to experiment with process control to get a better understanding of your process environment. You will identify the processes associated with your terminal session, work with variables in parent and child processes and terminate processes you have started.

Exercise Instructions

Preface

- All exercises for this unit depend on the availability of specific equipment in your classroom.

Structure

- ___ 1. Log in to the system and display your current process ID(PID).
- ___ 2. Create a subshell by entering `ksh`. What is the process ID of the subshell?
Is it different from your login process?
- ___ 3. Enter the command `ls -lR / > outfile 2> errfile &` and then execute the command which displays all of your running processes. The `ls` command will terminate when it finishes listing all the files in the directory tree.
- ___ 4. Terminate your child shell. What happens if you type `exit` from your login shell?

Process Environment

- ___ 5. Display all your variables that are in your current process environment.
- ___ 6. Create a variable `x` and set its value to 10. Check the value of the variable. Again, display all your current variables.
- ___ 7. Create a subshell with `ksh`. Check to see what value variable `x` holds in the subshell. What is the value of `x`? _____ List the subshell current variables. Do you see a listing for `x`? _____
- ___ 8. Return to your parent process. Set the value of variable `x` so that its value will be inherited by your child processes. Verify this by creating a subshell and checking on the value of variable `x`.
- ___ 9. Change the value of `x` to 200 in the subshell. Check that the value was changed.
- ___ 10. Go back to the parent process. Check on the value of `x` in this environment. Was the change in the subshell exported back to the parent?
- ___ 11. Create a shell script and name it **sc1**. It should read:
`pwd; cd /; pwd`
- ___ 12. Make the file **sc1** executable and run the program. What directory are you in now?

Why?

- ___ 13. Create another shell script and name it **sc2**. Have it read:
- ```
var1=hello; var2=$LOGNAME; export var1 var2
```
- \_\_\_ 14. Make **sc2** executable and run the program. When it is finished, examine the values of the variables `var1` and `var2`. What values do `var1` and `var2` have?

Why?

- \_\_\_ 15. Run the **sc2** program again, this time by forcing it to run in the current shell. When it is finished, check the values for `var1` and `var2`. What values do `var1` and `var2` have now?

Why?

## Job Control

- \_\_\_ 16. Create a shell script and name it **sc3**. It should read:

```
sleep 120
ls -lR / > outfile 2> errfile &
```

Make it executable. Start the script with the command:

```
$./sc3 > outfile 2> errfile
```

in the foreground.

- \_\_\_ 17. Suspend the job you just started.
- \_\_\_ 18. List all the jobs that you are running on the system and restart the above job in the background.
- \_\_\_ 19. Bring the job back to the foreground.
- \_\_\_ 20. Once the command finishes executing, restart it again in the background, display the process ID, and log off.
- \_\_\_ 21. Log in. Check to see if the process is still running.
- \_\_\_ 22. Start the **sc3** script with the **nohup** command, reference it using an explicit path and put it in the background. Do not forget to redirect the output from **sc3**, note its process ID and job number and then log off.
- \_\_\_ 23. Log in. Check to see if the process is still running. Hint: search for its process ID.
- \_\_\_ 24. When the process is complete, display the file that contains your output. (Hint: if you did not specify an output file, **nohup** will send the output to **nohup.out**.)
- \_\_\_ 25. Rerun the **sc3** script you just created placing it into the background but not using the **nohup** command. Note its process id and job number. Apply the **nohup** to the process ID of the background process **sc3** and then log off.

Log back into the system and verify that the process is still running.

### ***Terminating a Process***

- \_\_\_ 26. Use the `ls -lR /` command we have been using to start a long running job in the background. Note the process ID that is provided when you begin the background process. \_\_\_\_\_
- \_\_\_ 27. If you did not record the process ID when you first started the command in the background, how would you find it? \_\_\_\_\_

Once you know the process ID, kill the process. Check to be sure it was killed.

- \_\_\_ 28. Repeat instruction 26 above. Kill the process using the job number rather than the process ID. Check to be sure the job was killed.

### ***END OF EXERCISE***

# Exercise 10. Customizing the User Environment

## What This Exercise Is About

When users log in, they generally prefer their environment to be customized to meet their specific needs. In this exercise, the student will customize their environment with some very useful functions that are invoked every time they log in.

## What You Should Be Able to Do

After completing this exercise, students should be able to:

- Customize **.profile** and **.kshrc** files
- Set alias definitions

## Introduction

Half way through the exercise, you will change your primary prompt from a `$` to the name of your current directory. This changed prompt string will be reflected from that point on in the exercises. This will look different from what you are use to seeing in previous exercises.

If you are working in an X Windows session, when instructed to log out, execute one of the following commands: `$ su -` or `$ login`.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Customizing .profile and .kshrc*

- \_\_\_ 1. To customize your environment and have it take effect every time you log in, you must incorporate the changes in a file that is read at login. Ensure you are in your home directory. Edit your **.profile** file to add the following functions:
- 1) Change the primary prompt string to reflect the current directory.
  - 2) Display a message at login which contains your login name and the time you logged in.
  - 3) Define an alias named **dir** that invokes the **ls -l** command.
  - 4) Automatically set up the command line editing facility.
- \_\_\_ 2. Test your customization by re-executing your **.profile**. You can choose to log out and back in, or simply rerun it using the dot notation. Once you have done that, execute and answer the following:
- 1) Did your message display? \_\_\_\_\_
  - 2) Is your prompt the name of your home directory? \_\_\_\_\_
  - 3) Change to the **/etc** directory. Did your prompt change? \_\_\_\_\_
  - 4) Using **dir** do you get a long listing of your current directory? \_\_\_\_\_
  - 5) Invoke **dir** using command line editing.

If you answered NO to any question, edit your **.profile** and fix it.

- \_\_\_ 3. Once you have your customized **.profile** setup and functioning, open a subshell. Answer the following questions:
- i. Is your prompt the name of the current directory? \_\_\_\_\_
  - ii. Does the value of the alias **dir** still work? \_\_\_\_\_
  - iii. Can you invoke command line editing? \_\_\_\_\_
- \_\_\_ 4. Exit from the subshell and return to your home directory. Most settings, with the exception of system variables, only apply to the current environment and are not passed to subshells (child processes). To pass **alias** or **set** customized settings down to subshells, the **ENV** variable must be set in your **.profile** file along with the existence of a customized **.kshrc** file.

Revise your **.profile** and create the appropriate **.kshrc** file to support the **alias** and **set** customization you did in instruction 1.

In the **.profile** file, remove the 'alias dir' and 'set -o vi' customizations. Add the ENV variable assignment. Export both PS1 and ENV.

Add the **alias** and **set** customizations (you just removed from **.profile**) to **.kshrc**.

- \_\_\_ 5. Test your customization by re-executing your **.profile** file. Open a subshell and answer the following questions:
  - i. Is your prompt the name of the current directory? \_\_\_\_\_
  - ii. Is the value of the alias dir still working? \_\_\_\_\_
  - iii. Can you invoke command line editing? \_\_\_\_\_
- \_\_\_ 6. Exit the subshell and return to your login shell. Display a listing of all currently set alias names and locate the **dir** alias.
- \_\_\_ 7. Temporarily unalias **dir** without editing the **.kshrc** file. Then display the list of alias settings again and ensure that it is no longer defined. Try executing **dir**.
- \_\_\_ 8. The **dir** alias is still in your **.kshrc** file but is not set. The **unalias** command removed it from the list of current alias names. Invoke **.kshrc** to automatically add **dir** back in the alias list. Execute **dir**.

**END OF EXERCISE**





# Exercise 11. AIX Utilities (1)

## What This Exercise Is About

The purpose of this exercise is to become familiar with some of the most helpful data tools available with AIX.

## What You Should Be Able to Do

After completing this exercise, students should be able to execute recursive searches on directories for files that meet specific criteria.

## Introduction

This exercise is designed to give you experience using the **find** command.

Using the command line editing feature will be very helpful during this exercise as some of the commands can get quite lengthy and will be repeated in many instructions.

This exercise shows the `$` prompt; however, unless you reset the `PS1` variable from the prior exercise, you will see your current directory as your prompt.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *The find Command*

- \_\_\_ 1. Find and display all the files in the **/tmp** directory.
- \_\_\_ 2. Find all files in your home directory that begin with the letter **s** and have **ls -l** automatically execute on each file name found as a result of the search operation.
- \_\_\_ 3. Repeat the search in the previous step, but interactively prompt the user to display the long list on each file.
- \_\_\_ 4. Find all files starting from the **/usr** directory that are owned by the userid **uucp**. Modify the command line to count the number of files owned by **uucp**. There may be some directories that you do not have permission to read. This will cause a permission denied message to be displayed. Redirect all error messages to a file called **errfile**.
- \_\_\_ 5. Display the file **errfile** from the previous instruction to see if any errors messages were encountered.
- \_\_\_ 6. To demonstrate that **find** recursively searches all directories and subdirectories from the search path down, do the following:
  - a. Ensure you are in your home directory.
  - b. Make a subdirectory called **level1**.
  - c. Create a zero-length file named **letter1** in the subdirectory **level1**.
  - d. Change to the **level1** subdirectory.
  - e. Make a subdirectory under **level1** called **level2**.
  - f. Create a zero-length file named **letter2** in the subdirectory **level2**.
  - g. Change to your home directory.
  - h. From your home directory issue the command to list all files starting with the letter **l**. Record the names displayed.
  - i. From your home directory issue the command to **find** only files starting with the letter **l**. Record the names displayed.

***END OF EXERCISE***



# Exercise 12. AIX Utilities (2)

## What This Exercise Is About

The purpose of this exercise is to become familiar with some of the most helpful data tools available with AIX.

## What You Should Be Able to Do

After completing this exercise, students should be able to:

- Search text files for pattern matching
- Extract specific fields within a file
- Sort lines in a file
- Display the first or last few lines of a file.
- Log in to a remote system
- Transfer files between systems
- Save and restore files using the `tar` command

## Introduction

This exercise is designed to give you experience using some AIX data tools.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *The **grep** Command*

- \_\_\_ 1. Find all lines in the **/etc/passwd** file for user names that start with **team**.
- \_\_\_ 2. Find all lines in the **/etc/passwd** file that begin with the letter **t**.
- \_\_\_ 3. Find all lines in **/etc/passwd** that contain a digit 0–9.
- \_\_\_ 4. Repeat the search in the previous instruction, but this time display only the number of lines that contain the pattern.
- \_\_\_ 5. Use the **ps** and **grep** commands to display the processes initiated by users other than yourself, and pipe the output to the **more** command.

### *The **sort** Command*

- \_\_\_ 6. Display the content of the **/etc/passwd** file in alphabetic order. Next, display the contents of the file in reverse order.

### *The **head** and **tail** Command*

- \_\_\_ 7. Display the first 10 lines of **/etc/passwd**.
- \_\_\_ 8. Display the first 5 lines of **/etc/passwd**.
- \_\_\_ 9. Display the last 10 lines of **/etc/passwd**.
- \_\_\_ 10. The **tail** command is also handy for stripping out header information from the output of a command. First, list all processes currently running on your system. Notice the headings. Next, display all processes running on your system excluding the header information.

### *The **tn**, **ftp** and **tar** Commands*

- \_\_\_ 11. Log in to any remote system in your classroom. If you are not sure about the name of the remote system ask your instructor. Use one of the **teamxx** user IDs that have been supplied by your instructor.
- \_\_\_ 12. Execute the **hostname** command and verify that you really work on the remote system.
- \_\_\_ 13. Change to the **/tmp** directory and create a new file **testfile1**.
- \_\_\_ 14. Log out from the remote system.
- \_\_\_ 15. On your local system create a new directory **remote\_files**.

- \_\_\_ 16. Transfer the remote file **/tmp/testfile1** to your local system. The file should be stored in the subdirectory **remote\_files**.
- \_\_\_ 17. Verify that the file has been copied to your local system.
- \_\_\_ 18. Stay in the **remote\_dir** subdirectory and use the **tar** command to save all files in this directory. Create an archive file **/tmp/archive.tar** and save all files relatively.
- \_\_\_ 19. Verify the content of the archive file.
- \_\_\_ 20. Restore all files from your archive into the **/tmp**-directory.

**END OF EXERCISE**





## Exercise 13. AIX Utilities (3)

### What This Exercise Is About

This exercise allows you to experiment with additional helpful utilities that can be used in the AIX environment.

### What You Should Be Able to Do

After completing this exercise, students should be able to use `find`, `xargs`, and `file` to manipulate files.

### Introduction

You will be manipulating ordinary files and directories using commands discussed in lecture. Where there is more than one way to invoke a command, you will see an -OR- between the possible solutions in the **Exercise Instructions with Hints** section.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Using find, xargs, and file*

- \_\_\_ 1. Verify that you are in your `$HOME` directory. Create a subdirectory called **newdir**. Change to this directory and create five empty files in this directory with the names of **1**, **2**, **3**, **4**, and **5**.
- \_\_\_ 2. Show a simple list of the contents of the **newdir** directory. Now, list the contents of the **newdir** directory and pass the output to **xargs** to copy the files and rename them with the prefix **file** so the resulting copied file's name is **file1**, and so forth. Verify that the files were copied and the names assigned accordingly.
- \_\_\_ 3. Using **find**, **xargs**, and **grep**, display the names of the files under your home directory which contain the string **AIX**.
- \_\_\_ 4. Find out in which directory the **find** command is located. Determine the type of file (executable, ASCII, dir, etc) of the **find** command.
- \_\_\_ 5. Using the **find** command to recursively list the file under your home directory, determine the type of each file. This may be accomplished in two ways. You might first creating a file, named **myfiles** and then determine the file types of the files listed in **myfiles**. Alternatively you might do the entire task in a pipe, thus eliminating the need to create the file, **myfiles**.

### *END OF EXERCISE*

## Exercise 14. AIX Utilities (4)

### What This Exercise Is About

This exercise allows you to experiment with additional helpful utilities that can be used in the AIX environment.

### What You Should Be Able to Do

After completing this exercise, students should be able to:

- Use `diff`, `cmp`, and `dircmp` to compare files and directories
- Use `compress`, `zcat`, and `uncompress`
- Use `cat` to display non-printable characters

### Introduction

You will be manipulating ordinary files and directories using commands discussed in lecture. Where there is more than one way to invoke a command, you will see an -OR- between the possible solutions in the **Exercise Instructions with Hints** section.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Using `diff`, `cmp`, `dircmp`*

- \_\_\_ 1. Create a file called **list1**. In **list1**, list the names of several people you know, one line per name. Copy **list1** to a file called **list2**. Edit **list2** and make the following changes:
  - Change the spelling of one of the names.
  - Remove one of the names.
  - Add a new name.
- \_\_\_ 2. Using **`diff`**, compare the contents of **list1** and **list2**.
- \_\_\_ 3. Using **`cmp`**, compare the contents of **list1** and **list2**. Then invoke a complete or long comparison of the contents of both files.
- \_\_\_ 4. Using **`dircmp -d`**, compare your home directory with the home directory of another user account on your system (**teamyy**).

### *Using `compress`, `uncompress`, `zcat`*

- \_\_\_ 5. Copy the file **/etc/magic** to a file in your home directory named **mymagic**. Do a long listing on **mymagic** and record the number of bytes in the file: \_\_\_\_\_
- \_\_\_ 6. Using the verbose option with **`compress`**, compress **mymagic**. Record the percentage of compression, \_\_\_\_\_, and the name of the compressed file, \_\_\_\_\_. Do a long listing on the file and record the number of bytes. \_\_\_\_\_ Compare the number to the number in the previous instruction.
- \_\_\_ 7. Using **`zcat`**, expand and view the contents of **mymagic.Z**. You may want to page it as it is a large file.
- \_\_\_ 8. Using **`uncompress`**, restore the compressed file back to its original file. Invoke a long listing and record the number of bytes. \_\_\_\_\_ The number should be the same as the number in Step 5.

»

### *Displaying Non-Printable Characters*

- \_\_\_ 9. In your home directory, create a file named **invis** and type a few lines that include random tabs, spaces, **Ctrl-G**'s, and so forth, between the words. Display the file.

- \_\_\_ 10. Notice in the instruction above that when you displayed the contents of **invis** it did not look quite right. Display and locate all the non-printable characters to determine where you used spaces, tabs, control characters, and so forth.
- \_\_\_ 11. Create a directory named **invisdir** but insert an accidental **<Ctrl-g>** somewhere in the name.
- \_\_\_ 12. Invoke the following four commands. When asked to key in the **invisdir** name, do *NOT* enter the **<Ctrl-g>** you originally included as part of the name.
  - 1) Invoke a listing of files and directories in your home directory (**invisdir** should be included as part of the output).
  - 2) Try to invoke a long listing on the **invisdir** directory.
  - 3) Try to remove the **invisdir** directory.
  - 4) Repeat instruction a. above to see if there are any non-printable characters in the **invisdir** directory name that made instructions 2 and 3 fail.
- \_\_\_ 13. Using a method of your choice, successfully remove the **invisdir** directory.

***END OF EXERCISE***



# Exercise 15. Additional Shell Features

## What This Exercise Is About

After you have been using AIX for a while, you will find certain characteristics of your environment that you would like to customize along with some tasks that you execute regularly that you would like to automate.

This exercise will introduce you to some of the more common constructs used to help you write shell scripts in order to customize and automate your computing environment.

## What You Should Be Able to Do

After completing this exercise, students should be able to:

- List common constructs used in writing shell scripts
- Create and execute simple shell scripts

## Introduction

You need not have any programming experience to perform this exercise. Refer to the unit in the Student Notebook for help with the syntax of constructs when creating the shell scripts in this exercise.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Writing shell scripts*

- \_\_\_ 1. Create a shell script named **parameters** that will echo the five lines that follow using predefined special variables set by the shell to fill in the blanks. Execute the script using the positional parameters **10 100 1000**.

The name of this shell script is \_\_\_\_\_.  
The first parameter passed is number \_\_\_\_\_.  
The second parameter passed is number \_\_\_\_\_.  
The third parameter pass is number \_\_\_\_\_.  
Altogether there were \_\_\_\_\_ parameters passed.

- \_\_\_ 2. Using conditional execution, create a shell script named **checkfile** that will check to see if the file named **parameters** exists in your directory, and if it does, use a command to show the contents of the file. Execute the script.
- \_\_\_ 3. Modify the **checkfile** script and change the name of the file from **parameters** to **noname** (check to ensure that you do NOT have a file by this name in your current directory). Also, using conditional execution, if the **ls** command was NOT successful, display the error message, The file was not found. Execute the script. What else got displayed?
- \_\_\_ 4. Modify the **checkfile** script so that error messages from the **ls** command do not appear on the screen. Execute the script.
- \_\_\_ 5. Modify the **checkfile** script to accept a single parameter from the command line as input to the **ls** and **cat** commands. Execute the script twice, once using the file named **parameters** and again using the file named **noname**.

### *Using for, test, and if*

- \_\_\_ 6. Using the **for** loop, modify the **checkfile** script to accept multiple files as input from the command line instead of just one. If the files are found, display all of them. If the files are not found, display the error showing all file names that were not found. Look in your directory and jot down a few valid file names that you can use as input. Execute the script using valid and invalid file names.
- \_\_\_ 7. Change the **checkfile** script to use an **if** statement and **test** command rather than conditional execution to check if the **ls** command was successful. Execute the script as you did in the previous step. (Hint: Return codes play a part in this script.)



**Using `while` and `expr`**

- \_\_\_ 8. Create an endless **`while`** loop that will echo `Out to Lunch` every 5 seconds in a script named **`lunch`**. Execute the script. When you have seen enough, break the loop.
- \_\_\_ 9. From the command line, display the results of multiplying 5 and 6.
- \_\_\_ 10. Now using **`expr`**, create a shell script named **`math`** to multiply any two numbers when entered as input from the command line. Execute the script multiplying 5 times 6. Experiment with any other two numbers.

**END OF EXERCISE**



# Exercise 16. Using AIXwindows

## What This Exercise Is About

This exercise provides an opportunity to use AIXwindows.

## What You Should Be Able to Do

At the end of the lab, students should be able to:

- Start AIXwindows
- Manipulate screen windows using AIXwindows
- Open a new `aixterm` window
- Customize motif application on launch (optional)
- (optional) Use the `xhost` command and `DISPLAY` environment variable to execute an X Client on a remote system

## Introduction

It will be necessary to perform this machine exercise through a VNC session. Be sure to check with your instructor if you have any questions regarding the terminal you should use.

While in the AIXwindows environment, you may wish to minimize or close any windows not needed to prevent the terminal screen from becoming too cluttered.

This exercise also includes an optional exercise. Verify with your instructor that the machine setup will support the optional exercise.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Starting AIXWindows*

- \_\_\_ 1. Start your terminal emulator application, connect to the AIX system, and log in.
- \_\_\_ 2. Before we can start the VNC server application, you will need to set a VNC password for your remote session. Run the **vncpasswd** program, and when prompted, enter a password. Set your password to the same as your username (for example, if your userid was **team01**, set the password to **team01**).

»

- \_\_\_ 3. Start the VNC server application. Make note of the hostname/IP address and session number for the VNC server that is started.

Hostname: \_\_\_\_\_

IP Address: \_\_\_\_\_

Session number: \_\_\_\_\_

- \_\_\_ 4. Switch back to your lab workstation or lab portal facility and launch a VNC viewer. Enter in the **IPaddress:session** where appropriate, and the password. If all is correct, a window should appear with the AIXWindows environment in it, running under your assigned userid.

### *Working with Windows*

- \_\_\_ 5. Verify that the **aixterm** is the active window.
- \_\_\_ 6. Using the **aixterm**, try typing some AIX commands such as **ls**, **date**, **cal**, and **whoami**.
- \_\_\_ 7. Resize the width of the window.
- \_\_\_ 8. Change the height and width of the window, simultaneously.
- \_\_\_ 9. Drag the **aixterm** from one side of the screen to the other.
- \_\_\_ 10. Use the options in the window menu to move and resize the window.
- \_\_\_ 11. View the window menu again. Why do you think some items may be greyed out?
- \_\_\_ 12. Open the window menu on the **aixterm**, but now type the letter **m** rather than clicking **move**. Note that this is another way to move a window.

- \_\_\_ 13. The window menu also contains key sequence definitions (for example **Alt+F7**). These key bindings are known as accelerators.

What happens when you try pressing the **Alt+F7** key when the menu is posted?

What happens if you try a mnemonic when the menu is not posted?

- \_\_\_ 14. Iconify (minimize) the **aixterm** window. Once it is an icon, restore it back to the screen.
- \_\_\_ 15. Maximize the **aixterm** window. What happens? Once it is maximized, resize the window to a smaller size.

### ***Using the root Window***

- \_\_\_ 16. Use the root menu to open another **aixterm** window.
- \_\_\_ 17. Start another **xclock** from the root menu.

### ***Cut and Paste Functions***

- \_\_\_ 18. Within a window, use the vi editor to create a file called **tempfile**. Add a few lines of text to this file, but do not exit.
- \_\_\_ 19. Within a second **aixterm** window, use the vi editor to create another file called **tempfile.new**. Go into insert mode but do not add any text to this file at this time.
- \_\_\_ 20. Copy a few lines of text from **tempfile** in the first window to **tempfile.new** in the second window. When you have completed this step, exit vi in both windows.
- \_\_\_ 21. You have now completed this machine exercise. You may either try the optional steps that follow, end AIXwindows or lock your terminal.
- \_\_\_ 22. When you are done with working on this AIXWindows exercise, whether at this point or somewhere in the optional steps, be sure to go to the last step (step 28) and terminate the vncserver on your system.

### ***Command Line Options for aixterm (optional)***

- \_\_\_ 23. The **aixterm** command has many command line options. View these options using the **aixterm -help** command. You will need to pipe the output to **pg** or **more** as there is a lot of information.

If a printer is available, you may wish to print this information.

- \_\_\_ 24. Start an **aixterm** from the command line. Give the window the following characteristics:

**background color**    lightskyblue

**foreground color**    forestgreen

**font**                    rom10.iso1

**title**                    My Window

**full cursor**

**scrollbar**

Why do you think this window is smaller than the others?

- \_\_\_ 25. Start an **xclock** from the command line within one of the windows. Give the clock the following characteristics:

**background color**    white

**foreground color**    red

**hands on the dial**    blue

**second hand**            update every second

### ***The Client-Server Model (optional)***

- \_\_\_ 26. Use the **xhost** command to enable all other clients to access your X server.

You should see the message "Access control disabled, clients can connect from any host".

- \_\_\_ 27. Have another user (if possible, on a different server) try and start an **aixterm** and display it to your AIXWindows session. They will need to change the value of their `DISPLAY` variable to the hostname and X server number (which is the same as the VNC session number).

»

- \_\_\_ 28. In the new window that appeared, use the **id** command to verify that the window was started from the other user. Check the value of the `DISPLAY` variable. It should indicate the name of your host.
- \_\_\_ 29. From the remote system's window, execute the **xcalc &** command. From which system is the calculator being executed? You can verify this with the **ps** command. When you have completed this step, close the remote system's window.

- \_\_\_ 30. You have now completed the optional machine exercise. Shut down the VNC AIXWindows session from your original terminal emulator session.

***END OF EXERCISE***





# Exercise 17. Using the Common Desktop Environment (CDE)

## What This Exercise Is About

This exercise introduces you to the features of CDE.

## What You Should Be Able to Do

At the end of the lab, students should be able to:

- Recognize the various CDE controls on the Front Panel
- Use the Help Manager
- Start both an **ai~~x~~term** and **dtterm** terminal window
- Use the File Manager to navigate the directory structure, create new files (using the CDE text editor) and directories (folders), and place a file icon in the workspace backdrop
- Optionally, use the Calendar control to view the calendar, set appointments, and create reminders

## Introduction

This exercise is designed to provide an introduction to the features of CDE. You will use the Help Manager to obtain information as needed. Much of your work in this exercise will be with the File Manager, which is one of the most useful CDE functions.

If time permits, an optional exercise is included on the CDE calendar functions. Feel free to explore the other functions of CDE. In the next unit and exercise, you will learn to customize your CDE environment.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Exploring the Front Panel*

- \_\_\_ 1. Be sure that you have terminated the vncserver that was started for the Using AIXwindows exercise. The instructions for this were in the last step of that exercise
- \_\_\_ 2. If you are logging in on a locally connected graphics terminal, then enter your user information at the **Login Manager** panel.

If you are using VNC to create the CDE session, then connect to the AIX system using the ASCII terminal application and log in with your userid. Rename the existing **xstartup** file in the **\$HOME/.vnc** directory to xstartup.bak.

- \_\_\_ 3. Ensure that your present working directory is your home directory. Start the VNC server application. Make note of the hostname/IP address and session number for the VNC server that is started.

Hostname: \_\_\_\_\_

IP Address: \_\_\_\_\_

Session number: \_\_\_\_\_

- \_\_\_ 4. Switch back to your lab workstation or lab portal facility and launch a VNC viewer. Enter in the **IPaddress:session** where appropriate, and the password. If all is correct, a window should appear with the AIXWindows environment in it, running under your assigned userid.
- \_\_\_ 5. The CDE environment, by default, launches the Application Manager and File Manager. Close those two windows (we will restart them later).
- \_\_\_ 6. Locate the CDE Front Panel; you may need to scroll the desktop window to find it at the bottom of the window. Find the following components of the CDE Front Panel (do not click on them, just locate them):

Workspace Switch Buttons

Style Manager

File Manager

Application Manager

Personal Application Manager

Clock, Calendar

Mail

Trash Can

Exit Icon

Move Handles

Menu and Iconify Buttons

- \_\_\_ 7. Move the Front Panel to the top of the screen.
- \_\_\_ 8. Iconify the Front Panel and then restore it.

### ***Work with the Help Manager***

- \_\_\_ 9. From the Help subpanel note how options exist so that you can access AIX online documentation. The infocenter menu item will not work unless the infocenter facility has been configured on the lab system and has connectivity to an infocenter server. When you have reviewed the various Help functions, close the Help windows.

### ***Starting a Terminal Window***

- \_\_\_ 10. Start an **aixterm** Terminal Window.  

Now you have a terminal window where commands can be entered.
- \_\_\_ 11. Run some command line commands.
- \_\_\_ 12. Start the Desktop Terminal **dtterm**, using the Personal Applications Front Panel pop-up menu (the control that looks like a piece of paper and a pencil).  

The Personal Applications control is on the left side of the control panel, between the File Manager and Mail controls.
- \_\_\_ 13. Run some command line commands.
- \_\_\_ 14. Compare the **aixterm** and the **dtterm** windows. What differences do you see?
- \_\_\_ 15. In the **dtterm** session, use the **Edit** menu bar option to copy and paste text.
- \_\_\_ 16. Now close all open windows, except the Front Panel, and we will work with the File Manager.

### ***Working with the File Manager***

- \_\_\_ 17. Select the File Manager control from the Front Panel to access the File Manager.
- \_\_\_ 18. Make sure that you are in your Home Directory: called **/home/teamxx**. The current directory is displayed at the top of the window.
- \_\_\_ 19. View several of the files.
- \_\_\_ 20. So that you have a few items to work with, the first thing you'll need to do is create a few new files.
- \_\_\_ 21. Click on one of your new files and then click **Selected** in the menu bar. Click **Open** to edit the file. This will invoke the CDE Text Editor.

- \_\_\_ 22. Enter several lines of text in each file. Play with using the mouse pointer and/or cursor control keys to place your cursor in various locations in the text. Play with changing the text, using the insert and delete keyboard functions. You will notice that the CDE Text Editor is not the **vi** editor.

Content is immaterial, but for at least one of the files, create a small shell script.

When you have finished editing a file, save the file by clicking the **File** option in the menu bar, then selecting **Close**. Confirm that you want to save the file when that window is presented.

- \_\_\_ 23. Add execute permission to the shell script you just created. Once this is complete, execute the shell script.
- \_\_\_ 24. To execute the Shell Script, be sure its icon shows as a lightening bolt. Double-click the **Shell Script** icon. On the **Action:Run** window click **OK**. A window will appear showing the results of the Shell Script. Once you have reviewed the results, close the **Run** window.

Now you have a number of files that you can use in some drag and drop operations.

## ***Drag and Drop Operations***

You will need to be working with the files in your Home directory, so these should be displayed in the File Manager window.

If you are not at the correct directory, navigate up and down the structure until you get to where you want/ought/need to be.

- \_\_\_ 25. Use the mouse to move one of the files in your `$HOME` directory to the workspace backdrop. This will create a shortcut to access the file.

The file icon has been dropped onto the backdrop and will stay there for fast and convenient access. Now, if the file is executable, use the left mouse button and double-click the file icon to make it run.

- \_\_\_ 26. With the pointer on the file icon on the backdrop, press the right button on the mouse.

What actions can you take on the file?

You can drag a selected file from the Directory display presented by the File Manager or from the desktop and place it somewhere else. You cannot place the same file more than once on the desktop backdrop. You cannot drop a file on itself.

- \_\_\_ 27. While dragging a file, take it across the controls on the desktop.

What do you see?

- \_\_\_ 28. Drop the file on the Clock control. What happens?

## ***File Manager - Finding, Copying and Deleting Files***

The Desktop File Manager is one of the most useful and powerful tools in CDE. This section explores more of the File Manager capabilities.

- \_\_\_ 29. Set the File Manager preferences to display a Directory Tree diagram, starting at the root directory.
- \_\_\_ 30. Navigate to the root directory in the File Manager window.
- \_\_\_ 31. Expand the **/usr/dt** directory.
- \_\_\_ 32. Set your viewing options to see a single folder at a time (rather than a tree structure) and using small icons. Also request display of the full path using icons near the top of the window.
- \_\_\_ 33. Set your viewing options to display by properties (such as modify date, permissions, owner etc). This output will look similar to the output of the **ls -la** command.
- \_\_\_ 34. Close the File Manager and any windows that it opened.
- \_\_\_ 35. Use the File Manager to execute the **date** command. This command is found in the **/bin** directory.
- \_\_\_ 36. Use the File Manager to create the directory **cdelab** in your **\$HOME** directory.
- \_\_\_ 37. The File Manager can also be used to execute a find operation. Use the File Manager to find all pixmap files (files with an extension of **.pm**) in the CDE **/usr/dt** directory.  
     » .
- \_\_\_ 38. Copy two or more of the pixmap files to the **cdelab** subdirectory.
- \_\_\_ 39. Rename one of the files to **myicon.pm**.
- \_\_\_ 40. Delete the **myicon.pm** file using the mouse and the Front Panel trash can.
- \_\_\_ 41. Delete a second pixmap file using the File Manager Menu Bar.
- \_\_\_ 42. With CDE it is possible to retrieve a deleted file. Restore **myicon.pm**.
- \_\_\_ 43. Empty the trash can.
- \_\_\_ 44. Change the Owner and Group permissions of the restored file to read/write.
- \_\_\_ 45. Close the File Manager.
- \_\_\_ 46. At this point, you may continue with the optional exercise or exit out of CDE. If you are in a VNC environment, do not use the Exit icon, but instead just lock the session. If completely done with using the CDE interface go to the last step in this exercise and close down the interface.  
     Skip this step to perform the optional steps.

### Optional Exercise Steps

- \_\_\_ 47. Click the Calendar control on the Front Panel. Add an appointment in the next week.
- \_\_\_ 48. Change the view to Day View to view the appointment you have scheduled.

- \_\_\_ 49. Change the view to Week View to view the appointment you have scheduled.
- \_\_\_ 50. Set a reminder to yourself for the appointment. Make the appointment private so that others cannot view it on your calendar.
- \_\_\_ 51. Return to the *month view* icon on the calendar menu bar.
- \_\_\_ 52. Close the Calendar window
- \_\_\_ 53. Exit out of the vnc session.

***END OF EXERCISE***

# Appendix A. Customizing AIXwindows (1)

## What This Exercise Is About

This exercise shows the students how they can customize their AIXwindows environment.

## What You Should Be Able to Do

At the end of the lab, you should be able to:

- Customize the **.xinitrc** file
- Customize the **.Xdefaults** file

## Introduction

In this exercise, students will learn how to edit files to customize their AIXwindows environment.

## Exercise Instructions

### ***Preface***

- All exercises for this unit depend on the availability of specific equipment in your classroom.
- If VNC is used to provide the AIXWindows environment, then the file that controls the AIXWindows session initialization is called **xstartup**, and is located in **.vnc** subdirectory of the user's home directory.
- If the lab exercises is being done on a local attached graphics console (LFT), then follow the instructions marked **(LFT)**. If a VNC server session is being used to display AIXWindows, then follow the instructions marked **(VNC)**.

### ***Customizing the .xinitrc File***

The **.xinitrc** file is used by the **startx** shell script to initialize the AIXwindows session. (Actually, **startx** executes **xinit**, which reads the **.xinitrc** file.) If VNC is used to provide the AIXWindows environment, then the file that controls the AIXWindows session initialization is called **xstartup**, and is located in **.vnc** subdirectory of the user's home directory.

- \_\_\_ 1. Log in to your AIX system, either locally or through a terminal emulator.
- \_\_\_ 2. If using a local graphics display (LFT), copy the file **/usr/lpp/X11/defaults/xinitrc** into your **\$HOME** directory and call the file **.xinitrc**. If using a VNC session, there is no need to copy the file; the **xstartup** file is already present in **\$HOME/.vnc**.

»

- \_\_\_ 3. Edit the file and make the following changes:
- Add a second hand to the **xclock**.
  - Make the root window solid black.
  - Add, on a new line before the **exec mwm** line, the following:  
**aixterm -T "Bills Window" &**

»

- \_\_\_ 4. Start the AIXwindows session. If you are accessing AIXWindows through a VNC client, then start the VNC server with **vncserver**, and connect to the specified session with the client. Does the AIXwindows environment look different? It should!

»

### ***Customizing the .Xdefaults File***



- \_\_\_ 5. Execute the command **aixterm -keywords | pg** to view all the resources that can be customized for an **aixterm** window.
- \_\_\_ 6. Create the **.Xdefaults** file in your `$HOME` directory and add the following resource definitions:

```
Aixterm*foreground: DarkSlateGrey
Aixterm*background: wheat
Aixterm*geometry: 80x30
Aixterm*font: rom10.iso1
```

- \_\_\_ 7. Restart AIXwindows. This will cause your new **.Xdefaults** file to be read and used for any new **aixterm** windows you create. Now, open a new **aixterm** window. Does it have the characteristics specified in the **.Xdefaults** file?
- \_\_\_ 8. Now, end the AIXwindows session and then restart it. If using a VNC environment, switch to your ASCII terminal session and issue **vncserver -kill :session** where session is the VNC session ID. Restart the session by running **vncserver**. What do the two original windows look like? Why?

»

»

- \_\_\_ 9. Edit the **.Xdefaults** file and update the following lines for new colors:

```
Aixterm*foreground: grey
Aixterm*background: navy
```

- \_\_\_ 10. Restart the mwm and then create a new **aixterm** window from the command line. Does it use your new color specifications? It should!
- \_\_\_ 11. Exit your AIXwindows environment and log out from your system. If using VNC, switch to your ASCII terminal session and kill the VNC server.

»

## END OF EXERCISE



# Appendix B. Customizing AIXwindows (2)

## What This Exercise Is About

This exercise shows the students how they can customize their AIXwindows environment.

## What You Should Be Able to Do

At the end of the lab, you should be able to:

- Use the `custom` tool to tailor colors and fonts
- Use the `custom` tool to tailor size, location, icons, and the scrollbar
- Customize the Motif window manager (mwm)
- Use the `xsetroot` command to customize the root window

## Introduction

In this exercise, students will learn how to use the AIXwindows `custom` tool to customize their AIXwindows environment.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.
- If the lab exercises is being done on a local attached graphics console (LFT), then follow the instructions marked (**LFT**). If a VNC server session is being used to display AIXWindows, then follow the instructions marked (**VNC**).

### *Using the Custom Tool: Color and Fonts*

- \_\_\_ 1. Log in to your system and start AIXwindows. If the AIXWindows session is being accessed through VNC, start the VNC server, and access it from your VNC client.  
    »
- \_\_\_ 2. Make sure you have two **aixterm** windows open as well as the **xclock**. Also, start the scientific calculator.
- \_\_\_ 3. Start the AIXwindows customization tool.
- \_\_\_ 4. On the Customizing Tool window, choose **xcalc**.
- \_\_\_ 5. View the different resource categories that can be changed for the **xcalc** application. What sorts of resources can be changed? Choose **Colors**, which is the default resource category.
- \_\_\_ 6. Change the background color for **xcalc** to the color of your choice.
- \_\_\_ 7. Switch focus to an **aixterm** window and display the contents of **.Xdefaults**. Has it been updated? It should not have been!
- \_\_\_ 8. So, to have your values saved in **.Xdefaults**, change your focus back to the **xcalc** customizing window. Save the values you have chosen.
- \_\_\_ 9. Now, review the **.Xdefaults** file again. Your resource change should now be there.
- \_\_\_ 10. Return to the **xcalc** Customizing window and now choose the resource category of **Fonts**.
- \_\_\_ 11. View the various fonts that can be used for the `window interior`.
- \_\_\_ 12. The `List of Fonts` window is used to display all the possible fonts. Feel free to scroll through them, but be aware that there are LOTS of fonts in the list! You can narrow down the list of fonts by choosing **Family**, **Weight**, **Slant**, **Style**, **Spacing**, and **Size** in the respective selection windows. Below these windows will be feedback indicating how many fonts match the selection criteria.

Click a font from the **List of Fonts** that appears interesting. It will be displayed in the Sample box (some fonts will not display). If you have trouble finding a font you like, try the following to narrow down the search:

```
Family: Helvetica
Weight: Bold
Slant: All
Style: All
Spacing: All
Size: 14
```

Choose a font to be used for the **xcalc** window and save your choice as you did for the background color. Verify they change has been added to your **.Xdefaults** file. Close the Customizing windows.

- \_\_\_ 13. Use the customizing tool to change the background color for an **aixterm**. When you choose Apply will the color of your existing **aixterm** windows change like it did for the **xcalc** window? Will the new color be updated in the **.Xdefaults** file? Verify that your change updated **.Xdefaults** and affects the appearance of a new **aixterm**.

### ***Customizing the root Window with the xsetroot Command***

We will next change the root menu. This is done using the **xsetroot** command from the command line of one of your **aixterm** windows.

- \_\_\_ 14. Change the root window to solid blue.
- \_\_\_ 15. Change the cursor pointer to a skull and crossbones (called pirate), to a shuttle, or to gummy. Move the cursor to the root window to view the new cursor shape.
- \_\_\_ 16. Have the root window display **xsnow** (snowflakes) or **escherknots** - take your pick. These bitmap images are found in the directory **/usr/include/X11/bitmaps**. You may wish to view the file names in this directory for other bitmaps of interest. The bitmaps themselves are black and white images, so you may want to set other colors for the background and foreground.
- \_\_\_ 17. If you decide you like any of these root window options, how would you make your customization permanent, that is, available every time you start AIXwindows?

## Optional Exercises

### *Using the Custom Tool: Size and Location, Icons and Scrollbar*

- \_\_\_ 18. Make sure you have a running Calculator Tool. If not, start one.
- \_\_\_ 19. Start the AIXwindows Custom Tool and choose **xcalc** again.
- \_\_\_ 20. Choose the *Size and Location* resource category and customize the size of the **xcalc**.
- \_\_\_ 21. Suppose you wish to update the icon used for a particular AIXwindows application. To demonstrate how this is done, we will change the icon used for **xcalc**. You may first want to iconify and then restore the **xcalc** window to view the icon that is used. Then, use the **xcalc** Customizing window, and choose the icon resource category.
- \_\_\_ 22. Choose a new icon for the **xcalc** window: have the icon look like a terminal. Once you have completed this task, review the **.Xdefaults** file to verify that your entry has been added. Test the new icon to verify that it is being used.
- \_\_\_ 23. Now, add a scroll bar to the **aixterm** windows. Verify that the **.Xdefaults** file has been updated and test to verify that the scroll bar works.
- \_\_\_ 24. In your new **aixterm** window, list the files in **/usr/bin** and then use the scrollbar to go back and forth in the listing.

### *Customizing the Motif Window Manager (MWM)*

- \_\_\_ 25. Use the AIXwindows **custom** tool to update the MWM with the following characteristics:  

*window manager background: red*  
*window manager foreground: blue*

  
Verify that **.Xdefaults** has been updated.
- \_\_\_ 26. Some users prefer to use the pointer focus policy so they don't have to click a window to make it the active window. The pointer focus policy allows you to merely move the pointer to a window to make it the active window. If you are interested, change your focus policy to pointer. Verify that **.Xdefaults** has been updated and that the new focus policy works.

## **END OF EXERCISE**

# Appendix C. Customizing CDE

## What This Exercise Is About

This exercise provides an opportunity to customize the CDE Desktop.

## What You Should Be Able to Do

At the end of the lab, students should be able to:

- Customize CDE using the Style Manager
- Customize the Front Panel

## Introduction

Students will work as teams using a graphics terminal to customize their CDE environment. This machine exercise will focus on using the interactive customization features of CDE. First, the CDE environment will be customized using the Style Manager. Then, the Front Panel will be customized.

## Exercise Instructions

### *Preface*

- All exercises for this unit depend on the availability of specific equipment in your classroom.

### *Customizing the Front Panel*

- \_\_\_ 1. If logging in through a VNC client, first connect to the AIX machine via an ASCII terminal session and log in as your user. Start the VNC server session by typing **vncserver**. Switch back to your desktop, and start the VNC client application, specifying the hostname and VNC session number.

If logging in on a graphics console (LFT), log in as your userid.

»

»

- \_\_\_ 2. Customize your Workspaces as follows:

Rename each Workspace.

Change the Backdrop of each Workspace.

Turn on the screen saver and screen lock.

Set the window behavior.

Select a different palette for the workspaces.

- \_\_\_ 3. Add a fifth workspace and customize its style using the Style Manager.

- \_\_\_ 4. Set the new session as your Home session, and set Startup to return to your Home session at login.

**Note:** This is not supported in a VNC session. If using VNC, skip to step 6.

- \_\_\_ 5. Log out and log in again. Check to see that the state of your session matches what you set in the previous steps.

- \_\_\_ 6. Add the same **dtterm** session to all workspaces.

An application can be assigned to one or more workspaces by using the **Window** button menu.

- \_\_\_ 7. Use the **ls** command in the **dtterm** to list the current directory.

Check each of the workspaces to see if the same application session is available.

- \_\_\_ 8. Now, remove a Workspace application from one or more Workspaces.

- \_\_\_ 9. Have the **dtterm** application appear on the Front Panel as the default application associated with the **Personal Applications** control.



- \_\_\_ 10. Tear off the Personal Applications subpanel menu, and place it on the workspace.
- \_\_\_ 11. Create a new subpanel for the Style Manager control and add the Icon Editor and the **aixterm** applications to it.
- \_\_\_ 12. Now, remove the Icon Editor from the new subpanel.

### ***Adding a New Control to the Front Panel***

- \_\_\_ 13. Start the Application Manager.
- \_\_\_ 14. Open the Personal Applications subpanel.
- \_\_\_ 15. Drag the icon for **Firefox** from the Application Manager window on to the **Install Icon** from the Personal Applications subpanel.
- \_\_\_ 16. Close the Personal Applications subpanel.
- \_\_\_ 17. Find out the name of the definition file in directory **\$HOME/.dt/types/fp\_dynamic**. Write down the file name:
  
- \_\_\_ 18. Copy this definition file to directory **\$HOME/.dt/types** and specify a new file name.
- \_\_\_ 19. Anchor the application control in the Front Panel by editing the copied definition file. Use your student notes to find out which lines must be changed.
- \_\_\_ 20. Restart the CDE. After restarting CDE, you should see the application icon on the Front Panel.

### ***END OF EXERCISE***





