Assignments

Programming Assignments Information

Programming Assignment 1

Attached Files: Example output on Solaris (14.949 KB)
Example output on Ubuntu (14.291 KB)

Programming Assignment 1

Assignment 1

See Assignment / Project Requirements and Guidelines

Write a shell script to locate executable files. This script takes a list of file names from the command line and determines which would be executed had these names been given as commands.

- The search path should be based only on the user's PATH environment variable. **You shall not use** the Unix which command, the ksh whence (type)command, the locate command, or the bash type command.
- The code for the script shall not use the UNIX ls command to determine if the file is executable or if it exists.
- The script should find only the first occurrence of the "file". If the file is not found, the script the following

<command> NOT FOUND

<command> would be replaced by the name of the "file" you didn't find.

- If the first parameter is '-a', then the script should print all occurrences of the executable file in the user's path. Again if the file was not on the path, an error message should be displayed.
- **The find command shall not be used.**
- A project using temporary files will not be graded.

Note:

- The shell variable PATH defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list.
- If the command name contains a / then the search path is not used, you just check if the command/file specified is executable and not a directory. Otherwise, each directory in the path is searched for an executable file.

usage: mywhich [-a] command ...

Examples: The locations of these programs may vary on different systems and the users PATH environment variable.

prompt> mywhich ls
/bin/ls

prompt> mywhich -a cc
/bin/cc
/usr/ucb/cc

prompt> mywhich ./mywhich
./mywhich

prompt> mywhich /bin/ls
/bin/ls

prompt> mywhich fooblar
fooblar not found

prompt> mywhich ksh sh csh bash
/usr/bin/ksh
/bin/sh
/bin/csh
/usr/local/bin/bash

Programming Assignment 2

Attached Files: Example output and test cases (4.199 KB)

Programming Assignment 2

Assignment 2

See Assignment / Project Requirements and Guidelines

Write a shell script to concatenate lists together, and output the resulting list. Do not include any argument that is a sub-list of the entire list. (The script will clean the list of any redundant items.) You must preserve the original order of the list. Remember to account for the following situations:

- a :at the beginning of the list is the same as a . at the beginning of the "list" (:/bin is the same a .:/bin)
- a ::any where in the list is the same as :.: (/bin::/etc is the same as (/bin:.:/etc)
- a :at the end of the list is the same a :. (/bin:is the same as /bin:.)
- Project usings temporary files will not be graded.
- **The input to the script will be color or space separated lists and the output will be a colon separated list with the orignal order preserved and all redunant items removed.**

  USAGE: clean_list list ....
  Where list a a colon or whitespace separated list.

  Examples:

  prompt> clean_list a a:b a:b:c :x: y:z
  a:b:c:.:x:y:z

  prompt>clean_list /bin:/usr/bin:/usr/openwin/bin /usr/bin: /usr/etc:/etc: /usr/bin/X11 .:/bin
  /bin:/usr/bin:/usr/openwin/bin:/usr/etc:/etc:.:/usr/bin/X11

  prompt>clean_list apple:orange:apple pear orange peach
  apple:orange:pear:peach

REMEMBER TO HANDLE THE SPECIAL CASES OF LEADING :, A ::, AND A TRAILING :

Programming Assignment 3

Attached Files: student-example-p3 (2.732 KB)
student-example-p3.txt (2.868 KB)

Programming Assignment 3

Assignment 3

The student is to modify the script **student-example-p3** following the instructions in the script.

The purpose of this script is to change the name of files/directories with spaces in the name to have a dash(-). So if a file/directory were named "A B", the new name would be A-B. The script will work on a directory specified on the command line and replace the spaces in the names for all files/directories underneath that directory.

The options to the script are as follows:

-f the -f says to only rename files

-d the -d says to only rename directories

Both -d and and -f may be specifed.

<directory-name> the name of the directory that you will be processing.

For example:

Given the following

```
$ ls
C D/            dirs.tar        fix-spaces*
```

And the directory "C D" contained the following structure

```
./C D
./C D/E F
./C D/E F/G H
./C D/E F/G H/J    K
./C D/E F/G H/J    K/a b
./C D/E F/G H/a b
./C D/E F/a b
./C D/a b
```

After the script was run,

fix-space -f -d "C D"

the directory structure would be as follows:

```
C-D
C-D/E-F
C-D/E-F/G-H
C-D/E-F/G-H/a-b
C-D/E-F/G-H/J----K
C-D/E-F/G-H/J----K/a-b
C-D/E-F/a-b
C-D/a-b
```

Here is a copy of the script - it is alos as an attachment in both UNIX and Windows format.

```
#!/usr/bin/ksh

USAGE="$0 -f directory
$0 -d  directory
$0 -d -f directory

-f rename files
-d rename directories
"

usage ()
    {
    print -u2 "$USAGE"
    exit 1
```

```
    }

pathname ()
    {
    # function provided for the student
    print -- "${1%/*}"
    }

basename ()
    {
    # function provided for the student
    print -- "${1##*/}"
    }

find_dirs ()
    {
    # function provided for the student
    find "$1" -depth -type d -name '* *' -print
    }

find_files ()
    {
    # function provided for the student
    find "$1" -depth -type f -name '* *' -print
    }

my_rename()
    {
    # the student must implement this function to
my_rename
    # $1 to $2
    # The following error checking must happen:
    #    1. check if the directory where $1 resided is
writeable,
    #       if not then report an error
    #    2. check if "$2" exists -if it does report
and error and don't
    #       do the mv command
    #   3. check the status of the mv command and
report any errors
    : # remove this line after the function is coded
    }

fix_dirs ()
    {
    # The student must implement this function
    # to actually call the my_rename funtion to
    # change the name of the directory from having
```

```
spaces to
    # changing all of the spaces to -'s
    # if the name were "a b", the new name would be
a-b
    # if the name were "a   b" the new name would be
a----b
    : # remove this line after the function is coded
    }

fix_files ()
    {
    # The student must implement this function
    # to actually call the my_rename funtion to
    # change the name of the file from having spaces
to
    # changing all of the spaces to -'s
    # if the name were "a b", the new name would be
a-b
    # if the name were "a   b" the new name would be
a----b
    : # remove this line after the function is coded
    }

WFILE=
WDIR=
DIR=

if [ "$#" -eq 0 ]
   then
   usage
   fi

while [ $# -gt 0 ]
    do
    case $1 in
    -d)
        WDIR=1
        ;;
    -f)
        WFILE=1
        ;;
    -*)
        usage
        ;;
    *)
    if [ -d "$1" ]
        then
        DIR="$1"
```

```
    else
        print -u2 "$1 does not exist ..."
        exit 1
        fi
    ;;
    esac
    shift
    done

# The student must implement the following:
# - if the directory was not specified, the script
should
#   print a message and exit

# - if the Directory specified is the current
directory, the script
#   print a error message and exit

# - if the directory specified is . or .. the script
should print
#   an error message and exit

# - if both -f and -d are not specified, the script
should print a
#   message and exit
#

if [ "$WDIR" -a "$WFILE" ]
    then
    fix_files "$DIR"
    fix_dirs "$DIR"
elif [ "$WDIR" ]
    then
    fix_dirs "$DIR"
elif [ "$WFILE" ]
    then
    fix_files "$DIR"
    fi
```

Programming Assignment 4

Attached Files: Outline for Project 4 (1.243 KB)

Programming Assignment 4

Follow the my shell programming guidelines as in the other assignments.

Write a shell script to send a customized mail message to the users listed on the

command line by login (user)name, only if they are currently logged on.

- If no users are listed on the command line an error message should be printed.
- In the mail message, you should use the full (real)name from the passwd file (/etc/passwd).
- You also need to sign the script with the real name of the person who is running the script.
    - This can be derived from the $USER environment variable and looking up the value in the password file.
- An error message should be printed if the user does not exist in the passwd file.

- PLEASE ONLY SEND THIS MESSAGE TO USERS WHICH YOU KNOW PERSONALLY.
    - You can always use yourself and me (mrichard)as a test case
    - The real name of the user of the script should only be computed once.

- A "Here-Document (In-Line Redirection)" must be used for the mail message.
- No temporary files shall be used.

The message should be as follows

```
Hello <INSERT THE USERS REAL NAME FROM THE PASSWORD
FILE>,

**** This email is automatically generatated by
<username of the person running the script>  ******

My instructor requires that I send this message as
part of an assignment for class 92.312.
The current time and date is <insert the date/time
when the script is executing>.

Have a nice day.

<insert the real name of the person running the
script - do not hard code the value>
```