

Instructions

Complete the problems assigned below and turn in your answers by the due time above. To receive full credit, you must show both the relevant command(s) and output. If a question asks about what a command does, or what the output would be, give both the command and the output. Demonstrate that you actually performed the command, and didn't just guess.

Note – For the problems below, you do not need to worry about argument checking, error checking, or any the other requirements imposed on previous assignments. Just do exercises, focusing on solving the problem, not creating a shell program.

Homework Problems

1. Do problem 4 on page 312-313.

```
# sw function
#
# Changes directories to the directorires speicified by the first argument
# If arg is -, changes to previous working directory. If null, changes
# to $HOME
#
sw() {
    if [ $# -gt 1 ] ; then
        echo "sw: too many arguments\nUsage: sw [- | directory]"
        return 1
    fi

    current=`pwd`

    if [ "$1" == '-' ] ; then
        [ -n "$OLDPWD" ] && cd "$OLDPWD"
    elif [ -n "$1" ] ; then
        cd "$1"
    else
        cd
    fi

    OLDCWD="$current"
}
```

2. Write a simple program named **vars** that accepts one integer argument *N*, and creates *N* shell variables each assigned a different value, and print out the contents using the newly-created variable. The technique that must be used allows you to essentially create arrays in the shell. For example, **vars 20** would create the variables **var1**, **var2**, ... **var20**, and each will have a unique value assigned.

```
#!/bin/sh

# vars
#

prog=`basename $0`

# Syntax: Usage [ args ... ]
# Standard usage message and exit
Usage() {
    [ $# -ne 0 ] && echo $prog: $*
    echo usage: $prog n
    exit 1
}

[ $# -ne 1 ] && Usage "invalid number of arguments"
n=$1

# Syntax: Debug [ args ... ]
```

```

# Prints out debug message if DEBUG is set
Debug() {
    [ -n "$DEBUG" ] && echo "\tDebug: $"
}

while [ $n -gt 0 ] ; do
    n=`expr $n - 1`
    eval var$n=$n
    eval echo var$n = \${var$n}
done

```

3. Write a simple program that loops and on each iteration it increments an integer by one and sleeps for one second. The program outputs nothing until you type a Control-C character – then it outputs the current value of the counter. Control-C should not kill the program. Hint: you can use Control-backslash to kill the program (remove the core file afterwards).

```

#!/bin/sh

# traps
#

n=0

trap 'echo N is $n - signal 2 \((INT\) received' 2

while : ; do
    n=`expr $n + 1`
    sleep 1
done

```