**Department of Electrical and Computer Engineering**

Homework Assignment No. 03:

**Shell Programming**

submitted to:

Professor Joseph Picone

ECE 3822: Software Tools for Engineers

Temple University

College of Engineering

1947 North 12th Street

Philadelphia, Pennsylvania 19122

September 10, 2015

prepared by:

Robert Irwin  
Email: robert.irwin@temple.edu

# Problem

In this assignment we are programming our own mp3 player shuffle function in Shell.

# Approach

The first step of this problem was importing a list of .mp3 files to a directory called songs. A find command would have done this nicely, but I wanted to handpick my songs from a vast iTunes library. Because of this, I used “cp” to copy a .mp3 from iTunes to my songs directory. Because the names of the files contained spaces, they were not compatible with the mpg123 function I used to play the songs. To solve this problem, I ran the following line of code in my songs directory. This code replaces all spaces with an underscore.

find "." -name "\* \*" -type f | rename 's/ /\_/g' \*

Now that the filenames were dealt with, the code could start being written. The first step was creating an ordered list of numbers from one to the total amount of songs in the directory.

Because a shuffle function is random, a random number had to be generated. This was done by adding the current time in minutes and seconds together. Then we performed the following operation, SUM % SONGS\_LEFT+1. This ensures that our random number directly relates to a valid song number.

I then pulled the number on the nth line of the list of numbers and used that number to pick a song. After the song is picked, the number from the ordered list is deleted. This is how we ensure that no song can be played twice. For example, if the random number is 36, it goes to the ordered list and picks the 36th number, which in this case is 36. It then goes to the list of songs and plays the song on the 36th line. If the random number generator generates another 36, it goes to the 36th number of the ordered list, which is now 37. The 37th song of the song list is played. This also causes SONGS\_LEFT to be decremented by 1 every time a song is played. This ensures that our random number is less then or equal to the number of possible songs waiting to be played.

Then we had to ensure that our program could track itself so it would pick up where it left off if the program is restarted. Looking for the list of numbers our code generates did this. If it does not exist, it creates the full ordered list. If it does exist, it makes sure the list still has at least one number in it. If it does, it sets the count variable equal to the length of the list + 1. Count is the variable being used in modulus operation. The list of numbers is left alone. Based on the purpose of this list, it is evident that even after restarting the program, every song in the directory must be played once before any song can be played twice.

# Results

Roberts-MacBook-Pro:hw3 robertirwin$ rm num.list

Roberts-MacBook-Pro:hw3 robertirwin$ sh hw3.sh &

[1] 688

Roberts-MacBook-Pro:hw3 robertirwin$ CHECK = 0

Seconds: 04

Minutes: 07

Sum: 15

**Songnum: 15**

Songname: 02\_Hand\_Of\_Blood.mp3

**Count : 100**

High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3

version 1.22.4; written and copyright by Michael Hipp and others

free software (LGPL) without any warranty but with best wishes

Directory: songs/

Playing MPEG stream 1 of 1: 02\_Hand\_Of\_Blood.mp3 ...

MPEG 1.0 layer III, 192 kbit/s, 44100 Hz stereo

Title: Hand of Blood Artist: Bullet For My Valentine

Comment: 1 Album: Hand Of Blood

Year: 2005 Genre: Heavy Metal

**ps**

**PID TTY TIME CMD**

**393 ttys000 0:00.06 -bash**

**688 ttys000 0:00.01 sh hw3.sh**

**714 ttys000 0:00.29 mpg123 songs/02\_Hand\_Of\_Blood.mp3**

**Roberts-MacBook-Pro:hw3 robertirwin$ kill -9 714**

Roberts-MacBook-Pro:hw3 robertirwin$ hw3.sh: line 34: 714 Killed: 9 mpg123 songs/$songname

Seconds: 49

Minutes: 07

Sum: 5

**Songnum: 5**

Songname: 01\_-\_The\_Downfall\_Of\_Us\_All.mp3

**Count : 99**

High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3

version 1.22.4; written and copyright by Michael Hipp and others

free software (LGPL) without any warranty but with best wishes

Directory: songs/

Playing MPEG stream 1 of 1: 01\_-\_The\_Downfall\_Of\_Us\_All.mp3 ...

MPEG 1.0 layer III, 320 kbit/s, 44100 Hz stereo

Title: The Downfall Of Us All Artist: A Day To Remember

Comment:

Album: Homesick [Special Deluxe Edition]

Year: 2009 Genre: Post-Hardcore

ps

PID TTY TIME CMD

393 ttys000 0:00.06 -bash

**688 ttys000 0:00.02 sh hw3.sh**

732 ttys000 0:01.27 mpg123 songs/01\_-\_The\_Downfall\_Of\_Us\_All.mp3

**Roberts-MacBook-Pro:hw3 robertirwin$ kill -9 688**

Roberts-MacBook-Pro:hw3 robertirwin$ ps

PID TTY TIME CMD

393 ttys000 0:00.06 -bash

**732 ttys000 0:01.40 mpg123 songs/01\_-\_The\_Downfall\_Of\_Us\_All.mp3**

[1]+ Killed: 9 sh hw3.sh

**Roberts-MacBook-Pro:hw3 robertirwin$ kill -9 732**

**Roberts-MacBook-Pro:hw3 robertirwin$ sh hw3.sh &**

[1] 795

Roberts-MacBook-Pro:hw3 robertirwin$ CHECK = 1

Seconds: 55

Minutes: 12

Sum: 23

**Songnum: 25**

Songname: 03\_Fantasy\_Girl.mp3

**Count : 98**

High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3

version 1.22.4; written and copyright by Michael Hipp and others

free software (LGPL) without any warranty but with best wishes

Directory: songs/

Playing MPEG stream 1 of 1: 03\_Fantasy\_Girl.mp3 ...

MPEG 1.0 layer III, 320 kbit/s, 44100 Hz joint-stereo

Title: Fantasy Girl Artist: Man Overboard

Comment: This is a song. This song was uploaded by romaanx. It's awesome.

Album: Real Talk

Year: 2010 Genre: Pop Punk

Roberts-MacBook-Pro:hw3 robertirwin$ ps

PID TTY TIME CMD

393 ttys000 0:00.07 -bash

827 ttys000 0:00.76 mpg123 songs/03\_Fantasy\_Girl.mp3

[1]+ Killed: 9 sh hw3.sh

Roberts-MacBook-Pro:hw3 robertirwin$ kill -9 827

Roberts-MacBook-Pro:hw3 robertirwin$ more num.list

1

2

3

**4**

**6**

7

8

9

10

11

12

13

**14**

**16**

17

18

19

20

21

22

23

**24**

**26**

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

Roberts-MacBook-Pro:hw3 robertirwin$

From the output above, we can see that each song number that plays is removed from num.list which is why the random variable ‘sum’ does not always equal ‘songnum’. We can also see that both the list and COUNT pick up where the program left off after it is killed.

It is also important to note that the file num.list will not exist in a first-time user’s machine. Before the code was run initially, I removed num.list from my machine to ensure that the code will run without initial existence of num.list.

# Analysis

Here is the code used to complete the assignment.

#!/bin/sh

true=1

while [ $true ]

do

#We want to see how many songs we are working with

size="$(ls songs | wc -l)"

#We will keep track of our progress by grepping for num.list

#If num.list exists, we know the program has been run. We then print

#out the contents of the file and search it for any number. If we find a number,

#we leave num.list the way it is. We then set count to the length of the list + 1

ls > check.txt

check="$(grep 'num.list' check.txt | wc -l)"

echo "CHECK = " $check

if [ $check == 1 ]; then

check2="$(cat 'num.list' | grep -m 1 [0-9] | wc -l)"

if [ $check2 == 1 ]; then

num="$(cat 'num.list' | wc -l)"

COUNT=`expr $num + 1`

else

#Generate a list of ordered numbers from 1-#of\_files

seq 1 $size > num.list

#Add 1 because of modulo operation.

COUNT=`expr $size + 1`

fi

else

#Generate a list of ordered numbers from 1-#of\_files

seq 1 $size > num.list

#Add 1 because of modulo operation.

COUNT=`expr $size + 1`

fi

for i in {1..$size}

do

#Generate the random number

#This is done by modulating a number by 101, to ensure we get a number from 1-100

#COUNT is decremented each time through the loop.

#This is because the number of the song that was played is removed from

#num.list. Count is then decremented so the random number generated

#cannot exceed the amount of numbers left in the list.

#This is how we ensure that every song is played once before

#any song is played twice

sec="$(date +%S)"

min="$(date +%M)"

echo "Seconds: " $sec

echo "Minutes: " $min

SUM=`expr $sec + $min`

SUM1=`expr $SUM + $sec`

RAND=`expr $SUM1 % $COUNT`

#Make sure Rand doesn't equal 0

if [ $RAND == 0 ]; then

RAND=1

fi

echo "Sum: " $RAND

#Now we work wit the numbered list. This is how we assure that

#no song is played twice before every song is played once

#get the number of the song we want to play

#It is important to remeber that the song number we

#will play corresponds to the NUMBER on the line dictated

#by RAND.

songnum="$(head -$RAND num.list | tail -1)"

#get the song that corresponds to that number

songname="$(ls songs | head -$songnum | tail -1)"

echo "Songnum: " $songnum

echo "Songname: " $songname

COUNT=`expr $COUNT - 1`

echo "Count : " $COUNT

#delete the chosen number from the list of numbers so no song cant be played twice

sed "s/[[:<:]]$songnum[[:>:]]//" num.list | sed '/^\w\*$/d' > temp.list

cat temp.list > num.list

#Now we play the song

mpg123 songs/$songname

done

done

It is important to not that ps brings up all programs being run. If you look at the Results section of the report, you will find the command kill -9 SOME\_NUMBER. The number of the job you want to kill is made known with the ps command. To skip a song kill the mpg123 job. To kill the entire program, kill the .sh file first, and then the mpg123 job.

We want this to be run in the background. This is achieved by typing ‘**sh hw3.sh &**’on the command line.

This code is on Github at <https://github.com/robert-irwin/swtools_ece3822.git>.