**1032 Report – Semester 2 – 2024**

**Week 1:**

1. What CLI commands would you use to find five past commands you have used?

You would use ‘history 5’ as seen below:

A screen shot of a computer

Description automatically generated

It works by showing the 5 most recent commands, the command at the bottom (218) is the most recent and the top (214) is the 5th most recent command. If it was set to history 4 it would instead show the 4 most recent commands. If it was just history, it would show all commands used (in this case all 218).

**Week 2:**

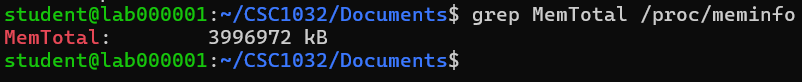
1.A. How to check using CLI what is the number of CPUs and size of the memory in the VM used during the practicals?

**To check the number of CPUs use ‘nproc –all’:**



This shows the amount of cores used in the virtual machine since ‘nproc -all’ prints the number of processing units in total. This returns 2.

**To check the size of memory use ‘grep MemTotal /proc/meminfo’ for just the total memory:**



This shows the total memory, if you wanted to get all information on the virtual machine’s memory you would use ‘/proc/meminfo’ by itself, ‘grep MemTotal’ picks out the single line which has ‘MemTotal’ in it as that’s the only relevant information for answering this question. This returns 3996972 kB.

1.B. How to check this information on Windows machines used in the lab (hint! task manager)?

**This information can be accessed through task manager by clicking on performance and then memory to get the size of memory or CPU to get the amount of cores:**

A screenshot of a computer

Description automatically generated

Here it shows the amount of available memory (5.3GB).

A screenshot of a computer

Description automatically generated

Here it shows the amount of CPU’s (cores), which is 14 in this case.

**Week 3:**

#!/bin/bash

echo "ID=C3021739"

echo "NAME=HollyWalker"

echo "DATE="$(date)

echo

if [ "$#" -ne 1 ]; then

echo "help: must have 1 parameter for example - './counter.sh [directory]'"

elif ! [[ -d "$1" ]]; then

echo "usage: parameter must be a directory for example - './counter.sh [directory]'"

else

directory=$1

echo "Counting words in files inside the directory " $directory

echo

count=0

echo "Total word count: "

find "$directory" -type f | xargs wc -w | tail -1

fi

In this script it checks the parameter given in by the user, if there is no parameter or more than one it will display the message "help: must have 1 parameter for example - './task10.sh [directory]'".

Only if there is a singular parameter it will proceed and check if the parameter is a directory, if not it displays the message "usage: parameter must be a directory for example - './task10.sh [directory]'".

If it is a directory, it will then go through each file contained within it and add up the cumulative word count, it will then output the total.

**Week 4/5/6:**

package main

import (

"bufio"

"fmt"

"os"

"os/exec"

"strings"

)

func execInput(input string) error {

input = strings.TrimSuffix(input,"\r\n")

input = strings.TrimSuffix(input,"\n")

args := strings.Split(input, " ")

switch args[0] {

case "exit":

os.Exit(0)

case "version":

fmt.Println("Go CLI v0.1 by Holly Walker (C3021739)")

return nil

}

cmd := exec.Command(args[0], args[1:]...)

cmd.Stderr = os.Stderr

cmd.Stdout = os.Stdout

return cmd.Run()

}

func main() {

hostname, err := os.Hostname()

if err != nil {

fmt.Fprintln(os.Stderr, err)

}

reader := bufio.NewReader(os.Stdin)

for {

fmt.Print("Hostname: Laptop-", hostname)

fmt.Print("> ")

input, err := reader.ReadString('\n')

if err != nil {

fmt.Fprintln(os.Stderr, err)

}

err = execInput(input)

if err != nil {

fmt.Fprintln(os.Stderr, err)

}

}

}

This is the go script for the shell CLI program, it’s role is to loop, continually taking input from the user until the user types the command “exit” where it will then cause the program to exit. Through this program the user can input system commands (both with and without arguments) alongside run the built in commands of “exit” and “version”.

If the user enters a command that isn’t accepted by the system, or a command that requires an argument when none is given, it will return an error message to the user, but will not cause the program to crash.

**Week 8:**

#include<math.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

//Holly Walker C3021739 (230217397)

//Quadratic equation function: takes the a,b,c values and calculates the roots

void quadraticEquation(float a, float b, float c)

{

  float root1,root2,real,imaginary;

  //Calculates discrimminant (this is to check the amount of roots/if they're imaginary)

  float discrimminant = (b\*b) - (4\*a\*c);

  printf("Enter either 'e' for exponential result or 'f' for float result\n");

  char dataType[2];

  scanf("%s", &dataType);

  if((strcmp(dataType,"e"))) {

    if (strcmp(dataType,"f")) {

      printf("Invalid input: must be either 'e' (exponential) or 'f' (float)");

      exit(0);

    }

  }

  //Roots are real and different

  if(discrimminant>0) {

    //Quadratic formula

    root1 = (-b + sqrt(discrimminant)) / (2\*a);

    root2 = (-b - sqrt(discrimminant)) / (2\*a);

    //float form

    if(!(strcmp(dataType,"f"))) {

      printf("root1 is %f \nroot2 is %f", root1, root2);

    }

    //exponential form

    else {

      printf("root1 is %e \nroot2 is %e", root1, root2);

    }

  }

  //Roots are real and the same

  if(discrimminant==0) {

    root1 = -b / (2\*a);

    root2 = root1;

    //float form

    if(!(strcmp(dataType,"f"))) {

      printf("root 1 and root2 is %f;", root1);

    }

    //exponential form

    else {

      printf("root 1 and root2 is %e;", root1);

    }

  }

  //Roots are imaginary and different

  if(discrimminant<0) {

    real =(-b) / (2.0\*a);

    imaginary = sqrt(-discrimminant) / (2\*a);

    //float form

    if(!(strcmp(dataType,"f"))) {

      printf("root 1 is %f+%fi \nroot 2 is %f-%fi", real, imaginary, real, imaginary);

    }

    //exponential form

    else{

      printf("root 1 is (%e)+%fi \nroot 2 is (%e)-%fi", real, imaginary, real, imaginary);

    }

  }

}

//Continually asks if the user would like to enter more quadratic equations

void mainLoop() {

  int exit = 0;

  while(exit == 0) {

    printf("\nWould you like to enter another quadratic equation? (type y for yes, n for no)\n");

    char answer[2];

    scanf("%s", &answer);

    if (strcmp(answer,"n")) {

      //if the answer is not n or y, an error message is shown

      if (strcmp(answer,"y")) {

        printf("Invalid input: must be either 'y' (yes) or 'n' (no)");

      }

      //if the answer is y, it prompts the user to enter a,b and c and then calls the quadratic equation

      else {

        printf("Enter a: \n");

        float a;

        scanf("%f", &a);

        printf("Enter b: \n");

        float b;

        scanf("%f", &b);

        printf("Enter c: \n");

        float c;

        scanf("%f", &c);

        quadraticEquation(a,b,c);

      }

    }

    //if the answer in n, the while loop exits

    else {

      exit = 1;

    }

  }

}

//Main program

void main( int argc, char \*argv[] )

{

  if(argc>1)

  {

    //If there's 3 arguments, it will use them as parameters for the quadraticEquation function

    if(argc==4)

    {

      float a = atof(argv[1]);

      float b = atof(argv[2]);

      float c = atof(argv[3]);

      quadraticEquation(a,b,c);

    }

    //Incorrect number of arguments prints error message

    else {

        printf("Invalid amount of arguments: either 3 arguments for values a,b and c or none(for manual input)");

    }

  }

  else {

    //If no arguments are entered in the command line, a b and c are manually entered by the user

    printf("Enter a: \n");

    float a;

    scanf("%f", &a);

    printf("Enter b: \n");

    float b;

    scanf("%f", &b);

    printf("Enter c: \n");

    float c;

    scanf("%f", &c);

    quadraticEquation(a,b,c);

  }

  mainLoop();

}

This code takes a quadratic equation in the format ax2 + bx + c = 0 and returns the quadratic roots. It does this by calling the quadraticEquation function, which first calculates the discriminant and then, depending on the discriminant’s value, it will calculate the roots. Once the roots are returned to the user, it will then call the mainLoop which will ask the user if they want to find the roots of more equations, if not the program will end, if they do the process will repeat again.

**Outputs with different coefficients**

**Float**

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Exponential**

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Week 9:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

//Holly Walker C3021739 (230217397)

//createArray function : gets input from the user, stores it into an array and returns the array

int\* createArray(int size) {

    int\* arr = calloc(size, sizeof(int));

    for (int i = 0; i < size; ++i) {

        printf("Enter an integer: \n");

        int x;

        scanf("%i", &x);

        arr[i] = x;

    }

    return arr;

}

//arraySearch function : searches through array for an element, and then prints it and it's position in the array

void arraySearch(int\* arr1, int arrSize1, int\* arr2, int arrSize2, int element) {

    //Searches through array 1

    for (int i = 0; i < arrSize1; ++i) {

        if (arr1[i] == element) {

            printf("\n%i was found in array 1 at postion %i",element,(i + 1));

        }

    }

    //Searches through array 2

    for (int i = 0; i < arrSize2; ++i) {

        if (arr2[i] == element) {

            printf("\n%i was found in array 2 at postion %i",element,(i + 1));

        }

    }

}

//printArraySum function : adds the elements in two arrays together and prints the results

void printArraySum(int\* arr1, int arrSize1, int\* arr2, int arrSize2) {

    //If array 1 is larger, the size of the result array will be the size of array 1

    if (arrSize1>arrSize2) {

        int\* arr = calloc(arrSize1, sizeof(arr[0]));

        for (int i = 0; i < arrSize1; ++i) {

            //If theres no more elements from array 2 left, the rest of the elements from array 1 are added

            if (i>arrSize2) {

                arr[i] = arr1[i];

            }

            //The elements at position i from array 1 and 2 are added together and stored in the result array

            else {

                arr[i] = arr1[i] + arr2[i];

            }

        }

        //Prints the result array

        printf("Array sum[%i] = {",arrSize1);

        for (int i = 0; i < arrSize1; i++)

        {

            if(i == (arrSize1-1)) {

                printf("%i", arr[i]);

            }

            else {

                printf("%i,", arr[i]);

            }

        }

        printf("}\n\n");

    }

    //If array 2 is larger, the size of the result array will be the size of array 2

    else {

        int\* arr = calloc(arrSize2, sizeof(int));

        for (int i = 0; i < arrSize2; ++i) {

            //If theres no more elements from array 1 left, the rest of the elements from array 2 are added

            if (i>arrSize1) {

                arr[i] = arr2[i];

            }

            //The elements at position i from array 1 and 2 are added together and stored in the result array

            else {

                arr[i] = arr1[i] + arr2[i];

            }

        }

        //Prints the result array

        printf("Array sum[%i] = {",arrSize2);

        for (int i = 0; i < arrSize2; i++)

        {

            if(i == (arrSize2-1)) {

                printf("%i", arr[i]);

            }

            else {

                printf("%i,", arr[i]);

            }

        }

        printf("}\n\n");

    }

}

//Main program

void main( int argc, char \*argv[] ) {

    //Takes size of array 1, then calls createArray function

    printf("Enter the size of array 1: \n");

    int arrSize1;

    scanf("%i", &arrSize1);

    int\* array1 = createArray(arrSize1);

    printf("\nIntegers for array 1 have been entered\n\n");

    //Takes size of array 2, then calls createArray function

    printf("Enter the size of array 2: \n");

    int arrSize2;

    scanf("%i", &arrSize2);

    int\* array2 = createArray(arrSize2);

    printf("\nIntegers for array 2 have been entered\n\n");

    printf("-----------------------------------\n");

    //Prompts user to choose between option a or b

    printf("Input 'a' if you'd like to complete operation a, or 'b' for operation b\n");

    char operation[2];

    scanf("%s", &operation);

    //If a is inputted, the printArraySum function is called

    if(!(strcmp(operation,"a"))) {

        printArraySum(array1, arrSize1, array2, arrSize2);

    }

    else {

        //If b is inputted, the printArraySum function is called

        if (!(strcmp(operation,"b"))) {

            printf("Enter the element to be searched for: \n");

            int element;

            scanf("%i", &element);

            arraySearch(array1,arrSize1,array2,arrSize2,element);

        }

        //If something other than a or b is inputted, an error message appears

        else {

            printf("Invalid input: must be either 'a' or 'b'");

        }

    }

}

This code first creates two arrays with the user’s input, it does this by getting the array size from the user and then calling the createArray function which gets each integer from the user and puts it into the array. Then it allows the user to choose between operation a and b. Operation a will call the printArraySum function which performs operation a as laid out in the specification. Operation b will get the element input from the user and then call arraySearch, which will then search through both arrays for the inputted element (as laid out in the specification for operation b).

**Outputs with different arrays**

**Array values of {1,1,1} and {1,1} where the element searched for is 1**

**OPERATION A**

A screenshot of a computer program

Description automatically generated

**OPERATION B**

A screenshot of a computer program

Description automatically generated

**Array values of {2,4,6,9} and {1,4,9,2,4,5} where the element searched for is 5**

**OPERATION A**

A screenshot of a computer program

Description automatically generated

**OPERATION B**

A screenshot of a computer program

Description automatically generated