Tên: Phan Thảo Vy-2252930

Question: Compare the Output Redirection (>/>>) with the Piping (|) technique.

- Output Redirection (>/>>) được sử dụng để chuyển đầu ra tới một tệp hoặc luồng. Điều này rất hữu ích để lưu kết quả đầu ra của lệnh để sử dụng hoặc phân tích sau này.
- Piping (|) cho phép đầu ra của một lệnh được gửi làm đầu vào cho lệnh khác yêu cầu. Điều này rất hữu ích để kết hợp nhiều lệnh và thực hiện các thao tác phức tạp trên đầu ra của một lệnh duy nhất.

Question: Compare the sudo and the su command.

- Công việc chính của lệnh su là cho phép bạn chuyển sang một số người dùng khác trong phiên đăng nhập. Nói cách khác, công cụ này cho phép bạn giả định danh tính của một số người dùng khác mà không cần phải đăng xuất rồi đăng nhập (với tư cách là người dùng đó).
- Lệnh su chủ yếu được sử dụng để chuyển sang tài khoản superuser/root (vì đặc quyền root thường được yêu cầu khi làm việc trên dòng lệnh), nhưng - như đã đề cập - bạn có thể sử dụng nó để chuyển sang bất kỳ người dùng không phải root nào khác.
- Lệnh "sudo" cho phép người dùng thực thi một lệnh duy nhất với các đặc quyền nâng cao, thường bằng cách nhập "sudo" trước lệnh. Người dùng được nhắc nhập mật khẩu riêng của họ và nếu được xác thực, lệnh sẽ được thực thi với các đặc quyền của người dùng được chỉ định, thường là người dùng siêu người dùng hoặc người dùng "root".

Question: Discuss about the 777 permission on critical services (web hostings, sensitive databases,...).

- Quyền 777 là một loại quyền tệp cấp quyền đọc, ghi và thực thi cho tất cả người dùng trên hệ thống. Đó là một cài đặt rất dễ dãi thường nên tránh, đặc biệt là trên các dịch vụ quan trọng như lưu trữ web hoặc cơ sở dữ liệu nhạy cảm.
- Khi quyền 777 được đặt trên một dịch vụ quan trọng, điều đó có nghĩa là bất kỳ người dùng nào trên hệ thống, bao gồm cả những kẻ tấn công tiềm năng,

đều có thể đọc, sửa đổi và thực thi các tệp được liên kết với dịch vụ đó. Điều này có thể dẫn đến các lỗ hổng bảo mật nghiêm trọng vì nó cho phép người dùng trái phép truy cập và có khả năng xâm phạm dữ liệu nhạy cảm.

- Dưới đây là một số lý do tại sao việc sử dụng quyền 777 trên các dịch vụ quan trọng lại gặp vấn đề:

+) Truy cập trái phép

+) Nâng cao đặc quyền

+) Tính toàn vẹn dữ liệu

+) Tuân thủ và quy định

Question:

1. What are the advantages of Makefile? Give examples?

2. Compiling a program in the first time usually takes a longer time in comparison with the next re-compiling. What is the reason?

3. Is there any Makefile mechanism for other programming languages? If it has, give an example?

1. Makefiles cung cấp một cách để tự động hóa quá trình xây dựng của một dự án, giúp việc biên dịch và quản lý các cơ sở mã lớn dễ dàng hơn. Chúng cũng cho phép theo dõi các phần phụ thuộc và chỉ biên dịch lại những phần cần thiết của dự án, cải thiện thời gian xây dựng tổng thể.

Ví dụ về Makefiles như tiện ích GNU Make và Makefile được sử dụng trong mã nguồn nhân Linux.

2. Lần đầu tiên chương trình được biên dịch, trình biên dịch cần tạo các tệp đối tượng và liên kết chúng với nhau để tạo ra tệp thực thi cuối cùng. Quá trình này có thể tốn thời gian, đặc biệt đối với các dự án lớn. Bên cạnh đó, ta còn có thể hiểu là ban đầu trình biên dịch cần phân tích, xử lý tất cả mã nguồn. Chẳng hạn như phân tích, xử lý từ vựng, ngữ pháp,...Ngoài ra, trình biên dịch có thể cần xây dựng các cấu trúc dữ liệu nội bộ và thực hiện các hoạt động tốn thời gian như liên kết các thư viện bên ngoài.

Tuy nhiên, việc biên dịch lại sau này chỉ cần cập nhật các phần của dự án đã thay đổi, nhanh hơn nhiều.

3. Có, có các cơ chế Makefile dành cho các ngôn ngữ lập trình khác, chẳng hạn như C++, Python và Java. Chúng thường sử dụng cú pháp tương tự như định dạng Makefile truyền thống nhưng với các công cụ và trình biên dịch dành riêng cho ngôn ngữ. Một ví dụ là công cụ CMake, được sử dụng để tạo Makefiles cho các dự án C++.

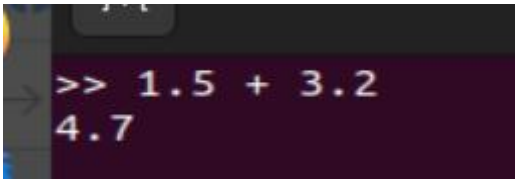
Bài tập 3.6

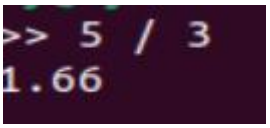
1. Đầu tiên, tạo một tệp bash (.sh) bằng `*touch*` và đặt quyền với `*chmod*

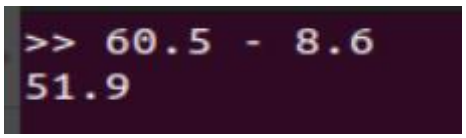
2. Chỉnh sửa tập tin bằng `*nano*` và nhấn `*Ctrl* + *X*` khi hoàn tất

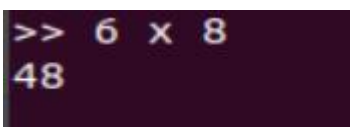
3. Chạy với *Bash*

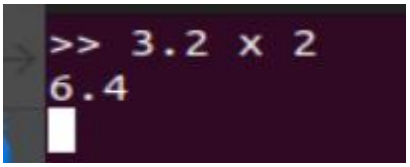
4. Ví dụ

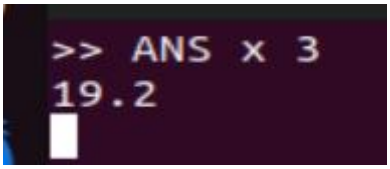


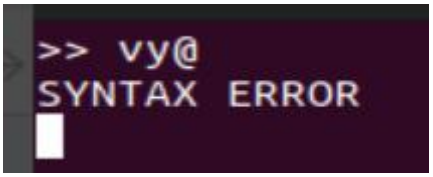


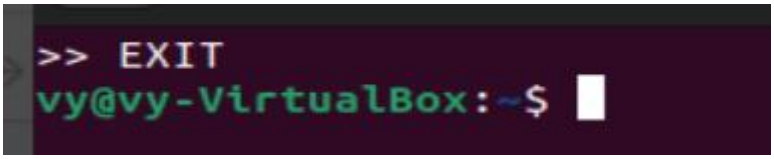












5. Source code

```
#! /bin/bash

declare -a hist
```

```
read -p ">> " num1 operator num2
```

```
if [ -f ans.txt ]
then
    read store < ans.txt
else
    store=0
fi
```

```
while [ "$num1" != "EXIT" ]
do
```

```
if [ "$num1" = "HIST" ]
then
    size=${#hist[@]}
    for ((i = 0; i <= $size; i++))
    do
        echo ${hist[$i]}
    done
elif [[ ! "$num1" =~ ^-?[0-9]*(\.[0-9]+)?$|ANS ]] || [[ ! "$num2" =~ ^-?[0-9]*(\.[0-
9]+)?$|ANS ]] || [[ ! "$operator" =~ [+x/%-] ]]
then
    echo -e "SYNTAX ERROR"
elif ([ "$operator" = "/" ] || [ "$operator" = "%" ]) && [ $num2 = 0 ]
then
    echo -e "MATH ERROR"
else
    ans_flag=-1
    if [ "$num1" = "ANS" ] && [ "$num2" = "ANS" ]
    then
        num1=$store
        num2=$store
        ans_flag=0
    fi
```

```
    if [ "$num1" = "ANS" ]
    then
```

```
        num1=$store
        ans_flag=1
    fi

    if [ "$num2" = "ANS" ]
    then
        num2=$store
        ans_flag=2
    fi

    case $operator in
    "+")res=`echo $num1 + $num2 | bc`
    ;;
    "-")res=`echo $num1 - $num2 | bc`
    ;;
    "x")res=`echo $num1 \* $num2 | bc`
    ;;
    "/")res=`echo "scale=2; $num1 / $num2" | bc`
    ;;
    "%")res=`echo "scale=0; $num1 / $num2" | bc`
    ;;
    esac
```

```
    if [[ ! "$res" =~ ^-?[0-9]+$ ]]
    then
        res=`printf  $res`
    fi
```

```
    case $ans_flag in
    0)num1=ANS
    num2=ANS
    ;;
    1)num1=ANS
    ;;
    2)num2=ANS
    ;;
    esac

    store=$res
    echo $store > ans.txt
```

```
    hist+=("$num1 $operator $num2 = $res")
```

```
    if [ ${#hist[@]} -gt 5 ]; then
        hist=("${hist[@]:1}")
    fi
    echo -e "$res"
fi
```

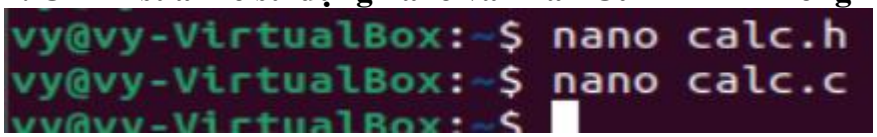
```
read -n 1
clear
read -p ">> " num1 operator num2
```

```
done
```

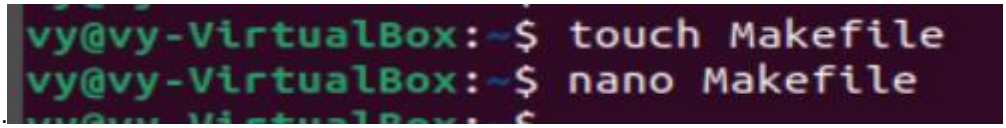
Bài tập 5.3

1. Đầu tiên, tạo tệp '.c' và '.h' bằng cách sử dụng `touch` và đặt quyền với `chmod`

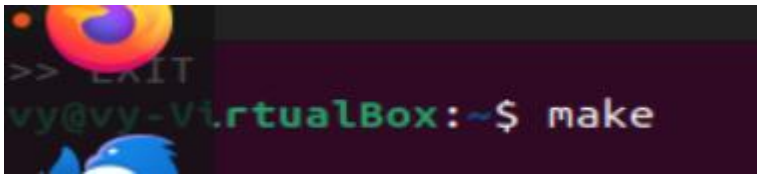
2. Chỉnh sửa file sử dụng nano và nhấn Ctrl + X khi xong



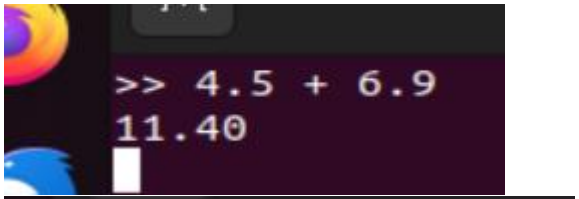
3. Tạo ra Makefile và chỉnh sửa với nano

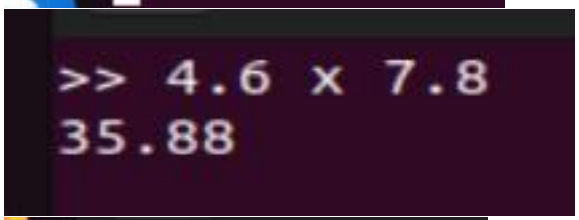


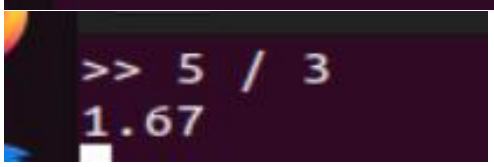
4. Chạy Makefile với lệnh make

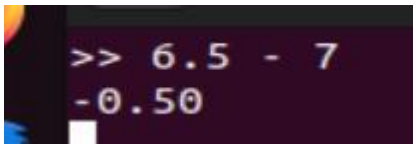


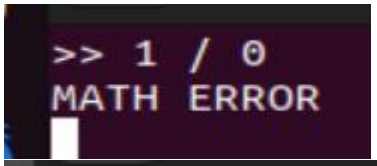
5. Kết quả

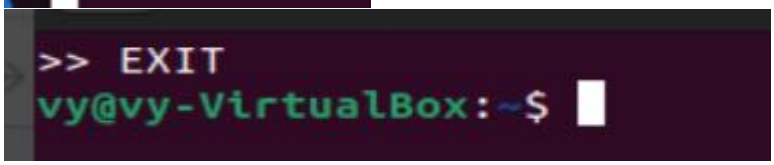












6. Source code

A. Code “calc.h”

```
/*******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite,
Prolog.
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CALC_H
float cong (float x, float y){
   return x+y;
}
float tru (float a, float b){
   return a-b;
}

float nhan (float a, float b){
   return a*b;
}

float chia (float z, float y)
{
   return z/y;
```

```
}
float mod(float a, float b){
   return (int) (a/b);
}
void main1 (){
   char chuoi[100];
   char *x, *y, *z;
   float p1, p2, ans = 0;
   while (1)
   {
      system("clear");
      printf(">> ");
      fgets(chuoi, 100, stdin);
      if (strcmp(chuoi , "EXIT\n") == 0)
      {
         break;
      }
      x = strtok(chuoi, " ");
      z = strtok(NULL, " ");
      y = strtok(NULL, " ");
      if (x == NULL || y == NULL || z == NULL)
      {
         printf("SYNTAX ERROR\n");
         getchar();
         continue;
      }
      if (strcmp(x, "ANS") == 0) // use strcmp to compare strings
         p1 = ans;
      else
      {
         p1 = strtod(x, NULL);
         if (p1 == 0 && *x != '0')
            {
               printf("SYNTAX ERROR\n");
               getchar();
               continue;
            }
      }
      if (strcmp(y, "ANS") == 0)
         p2 = ans;
      else
      {
         p2 = strtod(y, NULL);
         if (p2 == 0 && *y != '0')
         {
            printf("SYNTAX ERROR\n");
            getchar();
            continue;
         }
      }
      if (strcmp(z, "+") == 0)
         ans = cong(p1, p2);
      else if (strcmp(z, "-") == 0)
         ans = tru(p1, p2);
      else if (strcmp(z, "x") == 0)
         ans = nhan(p1, p2);
      else if (strcmp(z, "/") == 0)
      {
         if (p2 == 0)
         {
```

```
            printf("MATH ERROR\n");
            getchar();
            continue;
         }
         ans = chia(p1, p2);
      }
      else if (strcmp(z, "%") == 0)
         ans = mod(p1, p2);
      else
      {
         printf("SYNTAX ERROR\n");
         getchar();
         continue;
      }
      printf("%.2f\n", ans);
      getchar();
      }
 return ;
}
```

B. Code “calc.c”

```
#include "calc.h"
int main()
{
 main1();
 return 0;
}
```

C. Code Makefile

```
all:
	gcc -o calc calc.c -lm
	./calc
clean:
	rm -f calc
```