**BT 7.3 & 7.4**

Nguyễn Hải Dương - 20194530

**1. Bài tập 7.3:**



**Shell command execution**

Code:

#include <stdio.h>

#include <stdlib.h>

int main( )

{

  if(system("ls -l")==-1){

    perror("system");

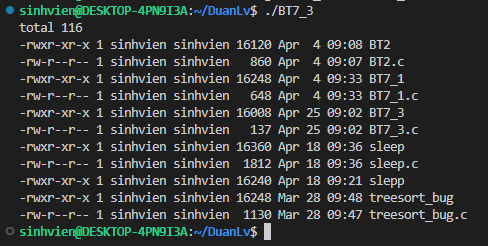
    return -1;

  }

  return 0;

}

- Kết quả thực thi chương trình:



- Giải thích:

Bằng cách sử dụng hàm system() để thực thi lệnh, chương trình in ra màn hình kết quả của thực thi lệnh đó vì thế chương trình hiển thị thông tin thư mục hiện tại là kết quả của lệnh “ls -l”

**Popen stream**

Code:

#include    <stdio.h>

#include    <string.h>

int main()

{

  char  str[512],\*ptr;

  FILE  \*fp;

  if((fp=popen("netstat","r"))==NULL){

    printf("popen error\n");

    return(-1);

  }

  while(1){

    fgets(str,512,fp);

    if(feof(fp)){

      break;

    }

    // \n is converted into \0.

    ptr=strchr(str,'\n');

    if(ptr!=NULL){

      \*ptr='\0';

    }

    printf("popen result   %s\n",str);

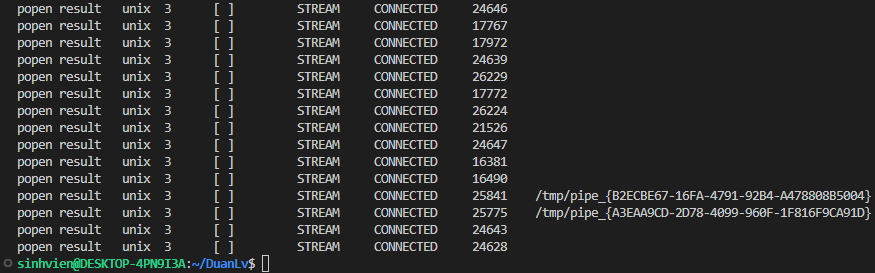
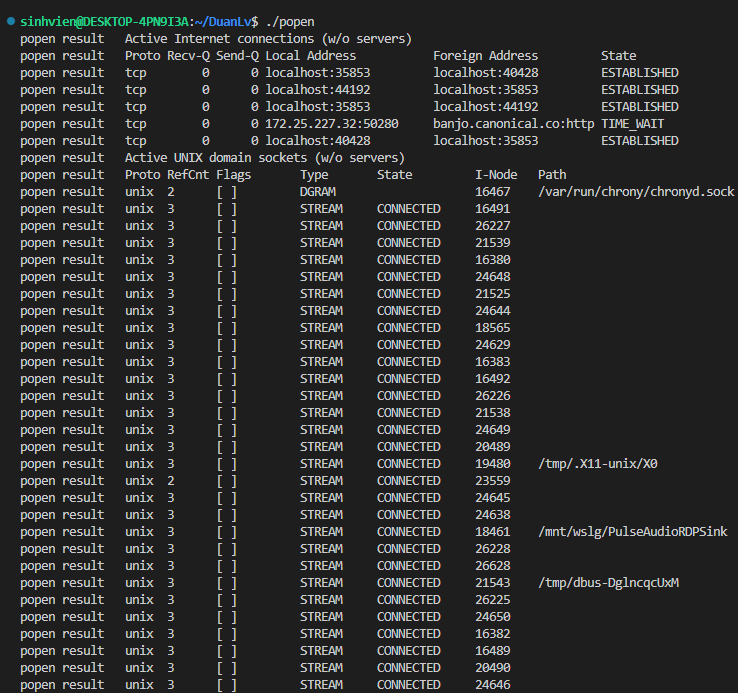
  }

  pclose(fp);

  return(0);

}

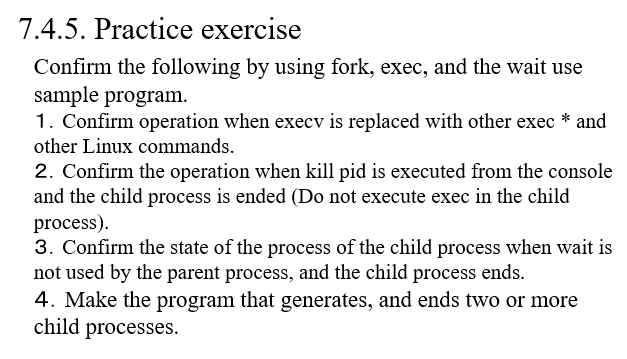
Kết quả thực thi chương trình:



- Giải thích:

Chương trình trên thu thập thông tin thống kê liên quan đến mạng bằng lệnh netstat bằng cách sử dụng hàm popen để mở một đường ống ghi kết quả thực thi câu lệnh vào file và hiển thị nội dung file ra ra màn hình.

**2. Bài tập 7.4:**



- Code:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#define PROC\_OK  0

#define PROC\_NG  -1

/\*\*

 \* The main processing

 \*/

int main(void)

{

  printf("main           : start\n");

  char \*args[] = {"ls", "-l", NULL};

  int iRet = 0;

  pid\_t pid = 0;

  int  status = 0;

  // Creating child process

  switch (pid = fork()){

    case -1:

      // Process creating error processing

      perror("processGenerate fork");

      exit(1);

    case 0:

      // Child process processing

      iRet = execv("/bin/ls", args);

      // It doesn't pass when execv succeeds.

      printf("execv ret = %d\n", iRet);

      if(iRet == PROC\_NG)

      {

        exit(1);

      }

    default:

      // Parent process processing

      printf("Parent Process ChildProcPid = %d\n", pid);

      // Wait for the state transition of the child process.

      pid=wait(&status);

      printf("pid=%d,status=%d\n", pid, status);

      break;

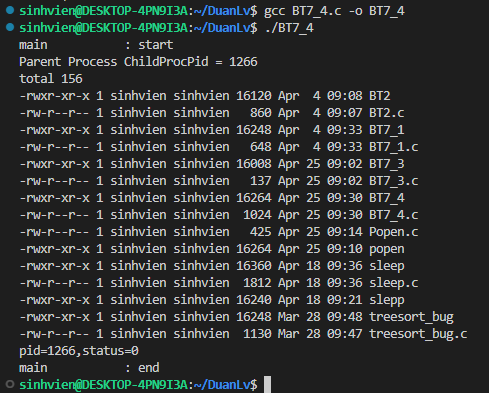
  }

  printf("main           : end\n");

  return 0;

}

- Kết quả thực thi chương trình:



Chương trình đợi chương trình con tạo, xử lý trong tiến trình con và kết thúc quá trình xử lý con bằng cách sử dụng fork, exec và wait.

・Nó đi vào trạng thái mà tiến trình con được tạo ra trong phần mà fork được thực thi, và hai tiến trình cha và tiến trình con được kết thúc.

・Bởi vì giá trị trả về sau khi fork được thực thi là 0, nên tiến trình con có thể được đánh giá.

Kết thúc quá trình sau khi thực hiện execv bằng cách xử lý trường hợp 0 ​​là phân nhánh của câu chuyển đổi và thực hiện lệnh ls.

・Tiến trình gốc là -1 có giá trị trả về sau khi fork được thực thi là một lỗi. Hoặc, có thể đánh giá nó vì không phải 0 đó là tiến trình con.

Đợi quá trình chuyển trạng thái của tiến trình con bằng cách đợi trong xử lý mặc định đó là sự phân nhánh trong mẫu cho chuyển đổi. Bắt đầu lại xử lý sau khi trạng thái của tiến trình con thay đổi và kết thúc chức năng chính.