Task1:

Design:

First fork the program

```
printf("Process start to fork\n");
pid = fork();
```

Check if error

```
if(pid==-1){
    perror("fork");
    exit(1);
}
```

Execute child

```
else{
    if(pid==0){
        int i;
        char *arg[argc];
        printf("I'm the Child Process, my pid = %d\n", getpid());
        printf("Child process start to execute test program:\n");
        for(i=0;i<argc-1;i++){
            arg[i] = argv[i+1];
        }
        arg[argc-1] = NULL;
        execve(arg[0],arg,NULL);
        perror("execve");
        exit(EXIT_FAILURE);
    }
```

Wait for the child program

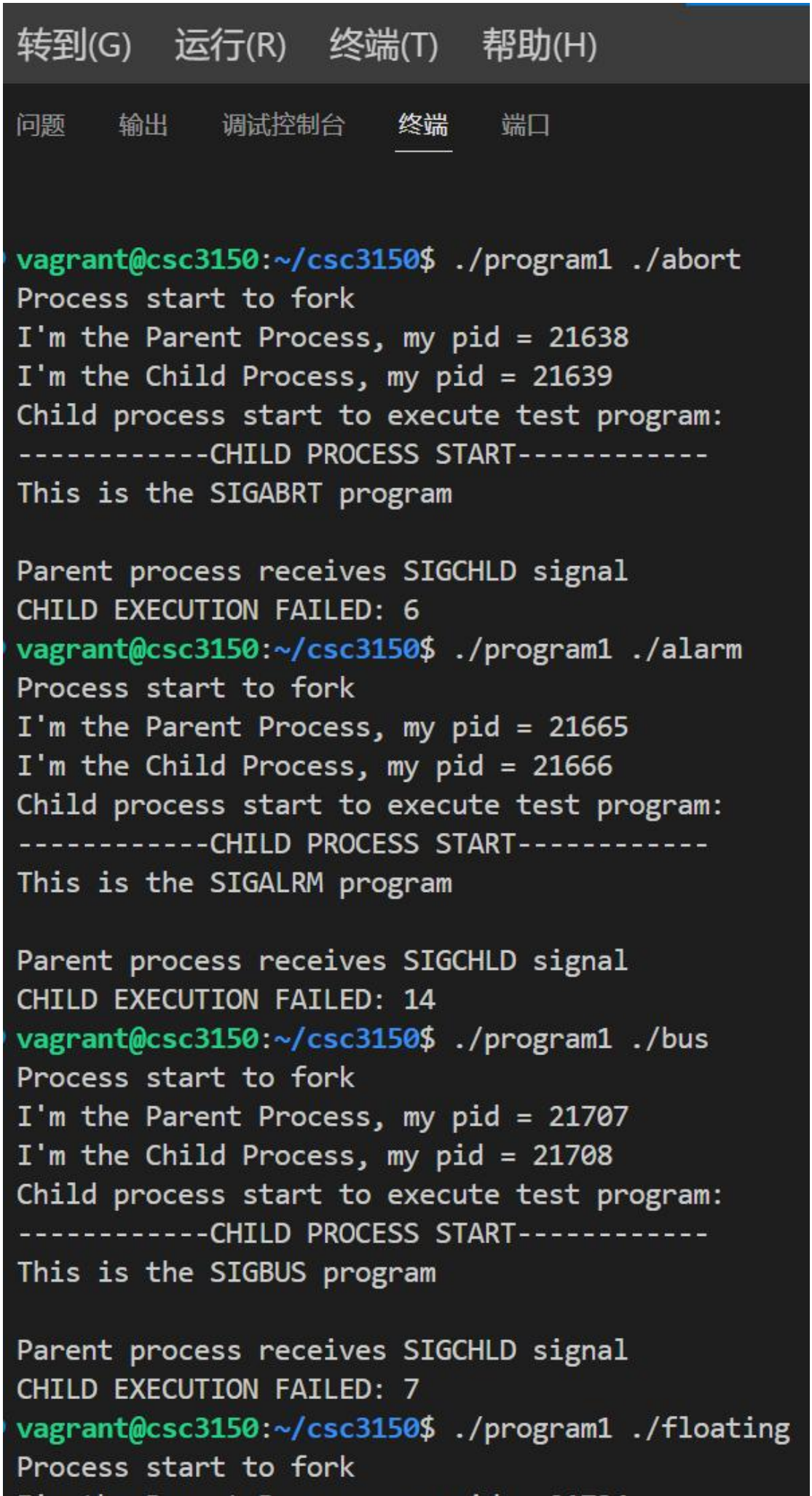
```
else{

    printf("I'm the Parent Process, my pid = %d\n", getpid());
    waitpid(pid,&status,WUNTRACED);
    printf("Parent process receives SIGCHLD signal\n");
```

Check the child's status

```
            if(WIFEXITED(status)){
                printf("Normal termination with EXIT STATUS = %d\n",WEXITSTATUS(status));
            }
            else if(WIFSIGNALED(status)){
                printf("CHILD EXECUTION FAILED: %d\n", WTERMSIG(status));
            }
            else if(WIFSTOPPED(status)){
                printf("CHILD PROCESS STOPPED: %d\n", WSTOPSIG(status));
            }
            else{
                printf("CHILD PROCESS CONTINUED\n");
            }
            exit(0);
        }
    }
    return 0;
}
```

Output


转到(G) 运行(R) 终端(T) 帮助(H)
问题 输出 调试控制台 终端 端口
vagrant@csc3150:~/csc3150$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 21638
I'm the Child Process, my pid = 21639
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program
Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 6
vagrant@csc3150:~/csc3150$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 21665
I'm the Child Process, my pid = 21666
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program
Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 14
vagrant@csc3150:~/csc3150$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 21707
I'm the Child Process, my pid = 21708
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program
Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 7
vagrant@csc3150:~/csc3150$ ./program1 ./floating
Process start to fork

```
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 1
vagrant@csc3150:~/csc3150$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 21809
I'm the Child Process, my pid = 21810
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 4
vagrant@csc3150:~/csc3150$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 21840
I'm the Child Process, my pid = 21841
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 2
vagrant@csc3150:~/csc3150$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 21879
I'm the Child Process, my pid = 21880
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 9
vagrant@csc3150:~/csc3150$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 21905
I'm the Child Process, my pid = 21906
Child process start to execute test program:
------------CHILD PROCESS START------------
```

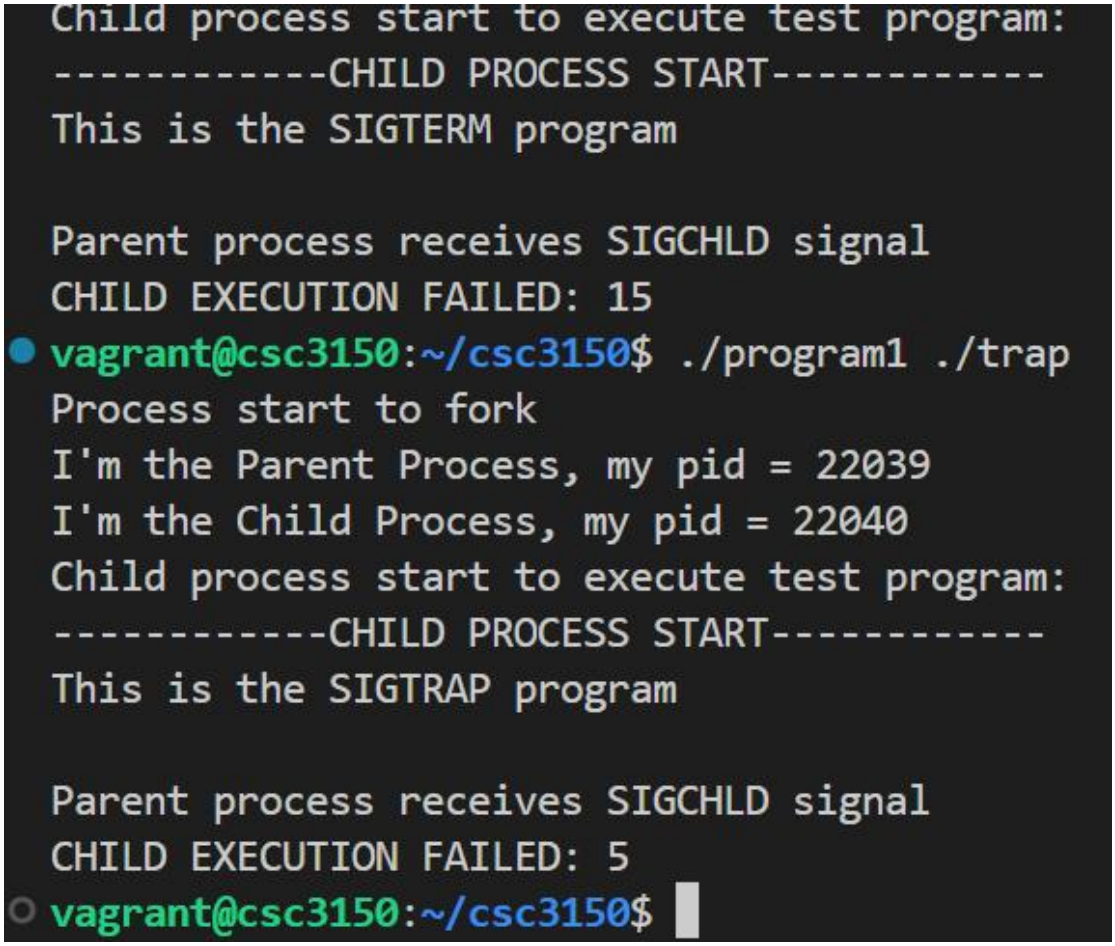
```
Normal termination with EXIT STATUS = 0
vagrant@csc3150:~/csc3150$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 21931
I'm the Child Process, my pid = 21932
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 13
vagrant@csc3150:~/csc3150$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 21945
I'm the Child Process, my pid = 21946
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 3
vagrant@csc3150:~/csc3150$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 21984
I'm the Child Process, my pid = 21985
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 11
vagrant@csc3150:~/csc3150$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 22008
I'm the Child Process, my pid = 22009
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED: 19
vagrant@csc3150:~/csc3150$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 22025
I'm the Child Process, my pid = 22026
```



Task2

Design

Execute program

```c
int my_exec(void){
    int c = 0;
    if(c==0);
    int output;
    char TestPath[] = "/tmp/test";
    struct filename *FileName = getname_kernel(TestPath);
    int a = 1;
    if(a==0);
    output = do_execve(FileName, NULL, NULL);
    int b = 2;
    if(b==0);
    if(output){
        do_exit(output);
    }
    else{
        return 0;
    }
}
```

Wait and get the child status

```c
void my_wait(pid_t pid, int status){
    int wat;
    struct wait_opts wo;
    struct pid * wo_pid = NULL;
    enum pid_type type = PIDTYPE_PID;
    wo_pid = find_get_pid(pid);

    wo.wo_type = type;
    wo.wo_pid = wo_pid;
    wo.wo_flags = WEXITED|WUNTRACED;
    wo.wo_info = NULL;
    wo.wo_rusage = NULL;
    // wo.wo_stat = status;

    wat = do_wait(&wo);
    if (wo.wo_stat == 0){
        printk("[program2] : This is a normal child process\n");
    }
    else if (wo.wo_stat == 1){
        printk("[program2] : get SIGHUP signal\n");
    }
    else if (wo.wo_stat == 2){
        printk("[program2] : get SIGINT signal\n");
    }
    else if (wo.wo_stat == 9){
        printk("[program2] : get SIGKILL signal\n");
    }
    else if (wo.wo_stat == 13){
        printk("[program2] : get SIGPIPE signal\n");
    }
    else if ( wo.wo_stat == 14){
        printk("[program2] : get SIGALRM signal\n");
    }
```

```
    printk("[program2] : get SIGTERM signal\n");
}
else if (wo.wo_stat == 131){
    printk("[program2] : get SIGQUIT signal\n");
}
else if (wo.wo_stat == 132){
    printk("[program2] : get SIGILL signal\n");
}
else if (wo.wo_stat == 133){
    printk("[program2] : get SIGTRAP signal\n");
}
else if (wo.wo_stat == 134){
    printk("[program2] : get SIGABRT signal\n");
}
else if (wo.wo_stat == 135){
    printk("[program2] : get SIGBUS signal\n");
}
else if (wo.wo_stat == 136){
    printk("[program2] : get SIGFPE signal\n");
}
else if (wo.wo_stat == 139){
    printk("[program2] : get SIGSEGV signal\n");
}
else if (wo.wo_stat == 4991){
    printk("[program2] : get SIGSTOP signal\n");
}
else{
    printk("[program2] : something other wrong\n");
}
printk("[program2] : child process terminates\n");
printk("[program2] : The return signal is %d\n", wo.wo_stat );
put_pid(wo_pid);
return;
```

Clone

```c
    /* fork a process using kernel_clone or kernel_thread */
    pid = kernel_clone(&c_args);
    int ppid = (int)current->pid;
    if(pid==-1){
        //perror("kernal_clone");
        //exit(1);
    }
    /* execute a test program in child process */
    else{
        if(pid==0){
            printk("[program2] : This is the parent process, pid =  %d\n",pid);
            my_exec();
        }

    /* wait until child process terminates */
        else{
            printk("[program2] : This is the parent process, pid =  %d\n", ppid);
            my_wait(pid,status);
        }

    }
    return 0;
}
```

About kernel

```c
struct wait_opts {
    enum pid_type wo_type;
    int wo_flags;
    struct pid *wo_pid;
    struct siginfo *wo_info;
    int wo_stat;
    struct rusage *wo_rusage;
    wait_queue_entry_t child_wait;
    int notask_error;
};

extern pid_t kernel_clone(struct kernel_clone_args *args);
extern struct filename *getname_kernel(const char * filename);
extern int do_execve(struct filename *filename,const char __user *const __user *__argv,const char __user *const __user *__envp);
static struct task_struct *task;
extern long do_wait(struct wait_opts *wo);
extern signed char my_WIFSIGNALED(int status);
```