CSC3150 Assignment1 Report

Part Ⅰ Program design

Program1:

1. Fork a pid for child process to execute the test program.
2. Use waitpid() to let parent process wait for child process's termination and receive the SIGCHLD signal from the child process.
3. Parent process handles the signal and prints out the termination status of the child process.

Program2:

1. Modify the kernel source code to export some useful functions I need.
2. Recompile the kernel.
3. Use kernel_clone to create a kernel thread and run my_fork() to fork a child process and use my_execve() execute the test file.
4. Use do_wait() to wait for the child process's termination.
5. After caught the signal from the child process, use some macros to handle it and print out the termination status of the child process in the kernel log.
6. Compile program2.c into a LKM and run it in the kernel mode.

bonus:

1. Use getopt() to get the parameters of the command.
2. Store each processes' infomation from /proc/(pid)/stat into a struct processNode and store the Nodes by a char[] .
3. Build the pstree (link the parent node and children nodes of each process).
4. Print the pstree in the terminate.
5. handle -p, -n, -V, -A, -l.

PartⅡ Development Environment

In this part I just follow the instructions in TUT2 and use make bzImage, make modules, make modules_install as well as make install to recomplie Linux 5.10.146. If any dependency packages are missing, install them.

PartⅢ Function Explanation

Program1:

1. Fork a child process

```
    /* fork a child process */
    pid_t pid;
    printf("Process start to fork\n");
    pid = fork();

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

2. Child process execute the test

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

3. Parent waits for the signal from the child process

```
else // Parent process
        {
            printf("I'm the Parent Process, my pid = %d\n" ,getpid());
            waitpid(-1,&status,WUNTRACED);
            if (WIFEXITED(status)) // The child process terminates normally
            {
                printf("Parent process receives SIGCHLD signal\n");
                printf("Normal termination with EXIT STATUS = %d\n",
WEXITSTATUS(status));
            }
            else if (WIFSIGNALED(status)) // The child process is terminated
by a signal
            {
                // int signal = WTERMSIG(status); // Get the signal code
that aborts the child process
                printf("Parent process receives SIGCHLD signal\n");
            }
            else if (WIFSTOPPED(status)) // Child process stopped
            {
                printf("Parent process receives SIGCHLD signal\n");
                printf("CHILD PROCESS STOPPED\n");
            }
            else
            {
```

```
                printf("Child PROCESS CONTINUE\n");
            }
            exit(0);
        }
```

Program2:

1. Function declaration

```
/*function declaration*/
int my_fork(void *argc);
int my_wait(pid_t pid);
static int __init program2_init(void);
static void __exit program2_exit(void);
int my_exec(void);
struct wait_opts;
```

2. Extern the syscall functions

```
/* extern the functions I need */
extern int do_execve(struct filename *filename, const char __user *const
__user *__argv, const char __user *const __user *__envp);
extern struct filename * getname(const char __user * filename);
extern struct filename * getname_kernel(const char * filename);
extern long do_wait(struct wait_opts *wo);
extern int kernel_wait(pid_t pid, int *stat);
extern pid_t kernel_clone(struct kernel_clone_args *args);
extern pid_t kernel_thread(int (*fn)(void *), void *arg, unsigned long
flags);
```

3. Fork some useful macros to handle the signals

```
int my_WEXITSTATUS(int status)
{
    return (((status) & 0xff00) >> 8);
}; //return exit status

int my_WTERMSIG(int status)
{
    return ((status) & 0x7f);
}; // return the signal number that terminated the process

int my_WSTOPSIG(int status)
{
    return my_WEXITSTATUS(status);
}; // return the signal number that terminated the process (19)

int my_WIFEXITED(int status)
{
    return (my_WTERMSIG(status) == 0);
}; // return true when the program exits normally

signed char my_WIFSIGNALED(int status)
{
```

```
    return (((signed char) (((status) & 0x7f) + 1) >> 1) > 0);
}; // return when the program terminates unnormally

int my_WIFSTOPPED(int status)
{
    return (((status) & 0xff) == 0x7f);
}; ///return true when a child process returns because it has been paused by
a SIGSTOP
```

4. Implement my_fork() (I handle the first 34 signals)

```
extern int my_fork(void *argc){
    //set default sigaction for current process
    int i;
    int status; // return value of my_wait()
    pid_t pid; // pid id for child process
    struct kernel_clone_args kca =
    {
        .flags = SIGCHLD,
        .child_tid = NULL,
        .parent_tid = NULL,
        .stack = (unsigned long)&my_exec,
        .stack_size = 0,
        .tls = 0,
        .exit_signal = SIGCHLD,
    };

    struct k_sigaction *k_action = &current->sighand->action[0]; //indicates
how the signal is handled
    for(i=0;i<_NSIG;i++){
        k_action->sa.sa_handler = SIG_DFL; // use the default handler
function
        k_action->sa.sa_flags = 0; //A flag specifying how the signal is to
be handled
        k_action->sa.sa_restorer = NULL;
        sigemptyset(&k_action->sa.sa_mask);
        k_action++;
    }

    /* fork a process using kernel_clone or kernel_thread */
    pid = kernel_clone(&kca); //fork a process
    printk("[program2] : The child process has pid = %d\n", pid);
    printk("[program2] : This is the parent process, pid = %d\n",
(int)current->pid);
    /* execute a test program in child process */

    /* wait until child process terminates */
    status = my_wait(pid);
    /* Process the returned signal */
    // printk("%d\n", status);
    if (my_WIFEXITED(status))
    {
        printk("[program2] : child process exit normally\n");
        printk("Normal termination with EXIT STATUS = %d\n",
my_WEXITSTATUS(status));
```

```
    } // exit normally
    else if (my_WIFSTOPPED(status))
    {
        // int sStatus = my_WSTOPSIG(status); // it should be 19
        printk("[program2] : child process get SIGSTOP signal\n");
    } // child process stop
    else if(my_WIFSIGNALED(status))
    {
        int tStatus = my_WTERMSIG(status);
        switch(tStatus) // I handle the first 34 signals here
        { ...
        };
        printk("[program2] : child process terminated\n");
    } //exit unnormally
    else
    {
        printk("[program2] : child process continue\n");
    } // continue
    printk("[program2] : The return signal is %d\n", my_WTERMSIG(status));
    do_exit(0);
    return 0;
}
```

5. implement my_wait()

```
int my_wait(pid_t pid){
    int a,b;
    struct wait_opts wo;
    struct pid *wo_pid = NULL;
    enum pid_type type;
    type = PIDTYPE_PID;
    wo_pid = find_get_pid(pid); //Look up a PID from hash table and return
with it's count evaluated

    wo.wo_type = type;
    wo.wo_pid = wo_pid;
    wo.wo_flags = WEXITED | WSTOPPED;
    wo.wo_info = NULL;
    wo.wo_stat = 0;
    wo.wo_rusage = NULL;

    a = do_wait(&wo);
    b = (wo.wo_stat);
    if (b == 0){
        return 0;
    }
    // output child process exit status
    put_pid(wo_pid); //Decrease the count and free memory

    return b;
}
```

5. implement my_exec()

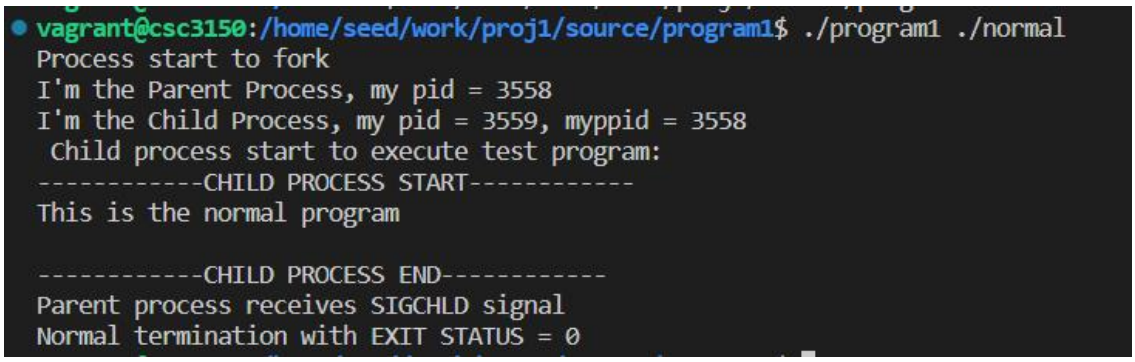
```c
int my_exec(void){
    int result;
    struct filename * myFilename =
getname_kernel("/home/seed/work/proj1/source/program2/test"); //get the exec
filename
    // struct filename * myFilename =
getname_kernel("/home/seed/work/proj1/source/program1/normal"); //get the
exec filename
    printk("[program2] : child process\n");
    result = do_execve(myFilename, NULL, NULL); //execute the task

    if(!result) //successfully execute
    {
        return 0;
    }
    else
    {
        printk("error code : %d\n",result); //print the error code
        do_exit(result);
    }
}
```

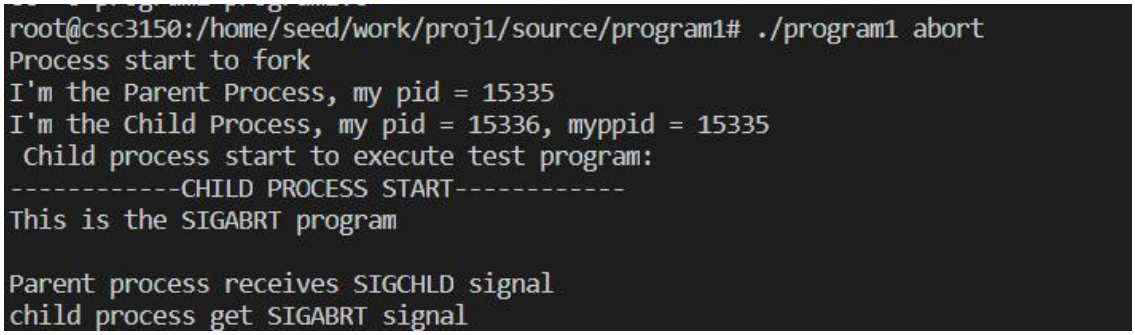
PartIV Screenshot of output

Program1:

1. Normal exit



2. Signaled abort



```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 alarm
Process start to fork
I'm the Parent Process, my pid = 15431
I'm the Child Process, my pid = 15432, myppid = 15431
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 bus
Process start to fork
I'm the Parent Process, my pid = 15522
I'm the Child Process, my pid = 15523, myppid = 15522
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process get SIGBUS signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 floating
Process start to fork
I'm the Parent Process, my pid = 15591
I'm the Child Process, my pid = 15592, myppid = 15591
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
child process get SIGFPE signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 hangup
Process start to fork
I'm the Parent Process, my pid = 15636
I'm the Child Process, my pid = 15637, myppid = 15636
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process get SIGHUP signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 kill
Process start to fork
I'm the Parent Process, my pid = 15919
I'm the Child Process, my pid = 15920, myppid = 15919
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
child process get SIGKILL signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 interrupt
Process start to fork
I'm the Parent Process, my pid = 15842
I'm the Child Process, my pid = 15843, myppid = 15842
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process get SIGINT signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 pipe
Process start to fork
I'm the Parent Process, my pid = 15942
I'm the Child Process, my pid = 15943, myppid = 15942
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process get SIGPIPE signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 3072
I'm the Child Process, my pid = 3073, myppid = 3072
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 segment_fault
Process start to fork
I'm the Parent Process, my pid = 16007
I'm the Child Process, my pid = 16008, myppid = 16007
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
child process get SIGSEGV signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 terminate
Process start to fork
I'm the Parent Process, my pid = 16075
I'm the Child Process, my pid = 16076, myppid = 16075
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
child process get SIGTERM signal
```

```
root@csc3150:/home/seed/work/proj1/source/program1# ./program1 trap
Process start to fork
I'm the Parent Process, my pid = 16119
I'm the Child Process, my pid = 16120, myppid = 16119
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
child process get SIGTRAP signal
```

3. Child process stopped

```
vagrant@csc3150:/home/seed/work/proj1/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 3632
I'm the Child Process, my pid = 3633, myppid = 3632
 Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED
```

Program2:

1. Normal exit

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[50291.687386] [program2] : module_init {Junxiao Liu} {120090809}
[50291.731199] [program2] : module_init create kthread start
[50291.753744] [program2] : module_init kthread start
[50291.788169] [program2] : The child process has pid = 6835
[50291.788209] [program2] : child process
[50291.815598] [program2] : This is the parent process, pid = 6834
[50291.852298] [program2] : child process exit normally
[50291.874859] Normal termination with EXIT STATUS = 0
[50291.920314] [program2] : The return signal is 0
[50295.229403] [program2] : Module_exit
```

2. Signaled abort

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[49989.698830] [program2] : module_init {Junxiao Liu} {120090809}
[49989.744372] [program2] : module_init create kthread start
[49989.766485] [program2] : module_init kthread start
[49989.804229] [program2] : The child process has pid = 5319
[49989.804259] [program2] : child process
[49989.807845] [program2] : This is the parent process, pid = 5318
[49991.851086] [program2] : get SIGALRM signal
[49991.885065] [program2] : child process terminated
[49991.924842] [program2] : The return signal is 14
[49994.910295] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1798.812125] [program2] : module_init {Junxiao Liu} {120090809}
[ 1798.812130] [program2] : module_init create kthread start
[ 1798.812415] [program2] : module_init kthread start
[ 1798.812530] [program2] : The child process has pid = 5256
[ 1798.812533] [program2] : This is the parent process, pid = 5255
[ 1798.812651] [program2] : child process
[ 1798.813552] [program2] : get SIGHUP signal
[ 1798.813556] [program2] : child process terminated
[ 1798.813557] [program2] : The return signal is 1
[ 1800.109693] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1874.204840] [program2] : module_init {Junxiao Liu} {120090809}
[ 1874.204845] [program2] : module_init create kthread start
[ 1874.205110] [program2] : module_init kthread start
[ 1874.205212] [program2] : The child process has pid = 5840
[ 1874.205234] [program2] : This is the parent process, pid = 5839
[ 1874.205370] [program2] : child process
[ 1874.408601] [program2] : get SIGILL signal
[ 1874.408605] [program2] : child process terminated
[ 1874.408607] [program2] : The return signal is 4
[ 1876.126644] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1962.418286] [program2] : module_init {Junxiao Liu} {120090809}
[ 1962.418290] [program2] : module_init create kthread start
[ 1962.418500] [program2] : module_init kthread start
[ 1962.418574] [program2] : The child process has pid = 6296
[ 1962.418577] [program2] : This is the parent process, pid = 6295
[ 1962.418621] [program2] : child process
[ 1962.419454] [program2] : get SIGINT signal
[ 1962.419459] [program2] : child process terminated
[ 1962.419461] [program2] : The return signal is 2
[ 1963.773245] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1996.029305] [program2] : module_init {Junxiao Liu} {120090809}
[ 1996.029309] [program2] : module_init create kthread start
[ 1996.029605] [program2] : module_init kthread start
[ 1996.029765] [program2] : The child process has pid = 6752
[ 1996.029794] [program2] : This is the parent process, pid = 6751
[ 1996.030015] [program2] : child process
[ 1996.030882] [program2] : get SIGKILL signal
[ 1996.030886] [program2] : child process terminated
[ 1996.030888] [program2] : The return signal is 9
[ 1998.384576] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1687.011061] [program2] : module_init {Junxiao Liu} {120090809}
[ 1687.011065] [program2] : module_init create kthread start
[ 1687.011274] [program2] : module_init kthread start
[ 1687.011397] [program2] : The child process has pid = 3845
[ 1687.011400] [program2] : This is the parent process, pid = 3844
[ 1687.011452] [program2] : child process
[ 1689.298477] [program2] : get SIGALRM signal
[ 1689.298480] [program2] : child process terminated
[ 1689.298481] [program2] : The return signal is 14
[ 1690.000083] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1732.821696] [program2] : module_init {Junxiao Liu} {120090809}
[ 1732.821699] [program2] : module_init create kthread start
[ 1732.821938] [program2] : module_init kthread start
[ 1732.822061] [program2] : The child process has pid = 4322
[ 1732.822064] [program2] : This is the parent process, pid = 4321
[ 1732.822223] [program2] : child process
[ 1733.035129] [program2] : get SIGBUS signal
[ 1733.035133] [program2] : child process terminated
[ 1733.035134] [program2] : The return signal is 7
[ 1734.495273] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 1765.274111] [program2] : module_init {Junxiao Liu} {120090809}
[ 1765.274116] [program2] : module_init create kthread start
[ 1765.274526] [program2] : module_init kthread start
[ 1765.274723] [program2] : The child process has pid = 4779
[ 1765.274726] [program2] : This is the parent process, pid = 4778
[ 1765.274810] [program2] : child process
[ 1765.476169] [program2] : get SIGFPE signal
[ 1765.476172] [program2] : child process terminated
[ 1765.476173] [program2] : The return signal is 8
[ 1766.632544] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 2110.504452] [program2] : module_init {Junxiao Liu} {120090809}
[ 2110.504456] [program2] : module_init create kthread start
[ 2110.504681] [program2] : module_init kthread start
[ 2110.504768] [program2] : The child process has pid = 7282
[ 2110.504771] [program2] : This is the parent process, pid = 7281
[ 2110.504837] [program2] : child process
[ 2110.505686] [program2] : get SIGPIPE signal
[ 2110.505692] [program2] : child process terminated
[ 2110.505694] [program2] : The return signal is 13
[ 2111.896705] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 2165.097598] [program2] : module_init {Junxiao Liu} {120090809}
[ 2165.097602] [program2] : module_init create kthread start
[ 2165.097779] [program2] : module_init kthread start
[ 2165.097871] [program2] : The child process has pid = 7744
[ 2165.097874] [program2] : This is the parent process, pid = 7743
[ 2165.097999] [program2] : child process
[ 2165.300870] [program2] : get SIGQUIT signal
[ 2165.300874] [program2] : child process terminated
[ 2165.300875] [program2] : The return signal is 3
[ 2166.379268] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 2196.430742] [program2] : module_init {Junxiao Liu} {120090809}
[ 2196.430745] [program2] : module_init create kthread start
[ 2196.430981] [program2] : module_init kthread start
[ 2196.431250] [program2] : The child process has pid = 8205
[ 2196.431254] [program2] : This is the parent process, pid = 8204
[ 2196.431351] [program2] : child process
[ 2196.643622] [program2] : get SIGSEGV signal
[ 2196.643625] [program2] : child process terminated
[ 2196.643626] [program2] : The return signal is 11
[ 2197.805202] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 2624.160599] [program2] : module_init {Junxiao Liu} {120090809}
[ 2624.160602] [program2] : module_init create kthread start
[ 2624.160801] [program2] : module_init kthread start
[ 2624.160915] [program2] : The child process has pid = 9138
[ 2624.160918] [program2] : This is the parent process, pid = 9137
[ 2624.160961] [program2] : child process
[ 2624.161975] [program2] : get SIGTERM signal
[ 2624.161980] [program2] : child process terminated
[ 2624.161982] [program2] : The return signal is 15
[ 2625.380577] [program2] : Module_exit
```

```
root@csc3150:/home/seed/work/proj1/source/program2# dmesg | tail -n 10
[ 2720.045410] [program2] : module_init {Junxiao Liu} {120090809}
[ 2720.045414] [program2] : module_init create kthread start
[ 2720.045640] [program2] : module_init kthread start
[ 2720.045813] [program2] : The child process has pid = 9595
[ 2720.045816] [program2] : This is the parent process, pid = 9594
[ 2720.045892] [program2] : child process
[ 2720.249204] [program2] : get SIGTRAP signal
[ 2720.249207] [program2] : child process terminated
[ 2720.249210] [program2] : The return signal is 5
[ 2721.317884] [program2] : Module_exit
```

3. Child process stopped

```
[  576.836057] [program2] : module_init {Junxiao Liu} {120090809}
[  576.836061] [program2] : module_init create kthread start
[  576.836251] [program2] : module_init kthread start
[  576.836362] [program2] : The child process has pid = 5331
[  576.836381] [program2] : This is the parent process, pid = 5330
[  576.836777] [program2] : child process
[  576.837547] [program2] : child process get SIGSTOP signal
[  576.837552] [program2] : The return signal is 19
```

Bonus:

1. pstree:

```
root@csc3150:/home/seed/work/proj1/source/bonus# make
gcc pstree.c -o pstree
root@csc3150:/home/seed/work/proj1/source/bonus# ./pstree
systemd-+-systemd-journal
        |-lvmetad
        |-dhclient
        |-iscsid
        |-iscsid
        |-dbus-daemon
        |-lxcfs
        |-rsyslogd
        |-accounts-daemon
        |-atd
        |-acpid
        |-cron
        |-sshd-+-sshd-+-sshd-+-bash-+-sh-+-node-+-node-+-bash-+-sudo-+-su-+-bash
                                                        |-bash
                                                        |-bash-+-sudo-+-su-+-bash-+-sudo-+-su-+-bash-+-pstree
                                                |-node-+-TabNine-+-TabNine-+-TabNine-deep-cl
                                                                          |-WD-TabNine
                                                        |-node
                                                        |-node
                                                        |-cpptools
                                                |-node
                                        |-sleep
        |-mdadm
        |-unattended-upgr
        |-polkitd
        |-irqbalance
        |-agetty
        |-login-+-bash
        |-systemd-+-(sd-pam)
        |-sudo-+-su-+-bash
        |-sudo-+-su-+-bash
        |-systemd-udevd
        |-systemd-logind
        |-cpptools-srv
        |-cpptools-srv
```

2. pstree -p

```
root@csc3150:/home/seed/work/proj1/source/bonus# ./pstree -p
systemd(1)-+-systemd-journal(557)
           |-lvmetad(587)
           |-dhclient(1047)
           |-iscsid(1195)
           |-iscsid(1196)
           |-dbus-daemon(1201)
           |-lxcfs(1202)
           |-rsyslogd(1207)
           |-accounts-daemon(1215)
           |-atd(1227)
           |-acpid(1231)
           |-cron(1252)
           |-sshd(1260)-+-sshd(1729)-+-sshd(1764)-+-bash(1765)-+-sh(1810)-+-node(1820)-+-node(1906)-+-bash(4156)-+-sudo(4180)-+-su(4181)-+-bash(4182)
                                                                                                                  |-bash(8762)
                                                                                                                  |-bash(11842)-+-sudo(11994)-+-su(11995)-+-bash(11996)-+-sudo(13226)-+-su(13227)-+-bash(13228)-+-p
stree(15352)
                                                                                                       |-node(1939)-+-TabNine(2007)-+-TabNine(2085)-+-TabNine-deep-cl(2941)
                                                                                                                                    |-WD-TabNine(12307)
                                                                                                                  |-node(2317)
                                                                                                                  |-node(8049)
                                                                                                                  |-cpptools(12760)
                                                                                                       |-node(1950)
                                                                                             |-sleep(15321)
           |-mdadm(1284)
           |-unattended-upgr(1323)
           |-polkitd(1325)
           |-irqbalance(1367)
           |-agetty(1369)
           |-login(1372)-+-bash(1703)
           |-systemd(1398)-+-(sd-pam)(1399)
           |-sudo(7140)-+-su(7141)-+-bash(7142)
           |-sudo(8063)-+-su(8064)-+-bash(8065)
           |-systemd-udevd(12298)
           |-systemd-logind(12874)
           |-cpptools-srv(13041)
           |-cpptools-srv(13101)
```

3. pstree -n

```
root@csc3150:/home/seed/work/proj1/source/bonus# ./pstree -n
systemd-+-systemd-journal
        |-lvmetad
        |-dhclient
        |-iscsid
        |-iscsid
        |-dbus-daemon
        |-lxcfs
        |-rsyslogd
        |-accounts-daemon
        |-atd
        |-acpid
        |-cron
        |-sshd-+-sshd-+-sshd-+-bash-+-sh-+-node-+-node-+-bash-+-sudo-+-su-+-bash
                                                              |-bash
                                                              |-bash-+-sudo-+-su-+-bash-+-sudo-+-su-+-bash-+-pstree
                                                    |-node-+-TabNine-+-TabNine-+-TabNine-deep-cl
                                                                             |-WD-TabNine
                                                              |-node
                                                              |-node
                                                              |-cpptools
                                                    |-node
                                          |-sleep
        |-mdadm
        |-unattended-upgr
        |-polkitd
        |-irqbalance
        |-agetty
        |-login-+-bash
        |-systemd-+-(sd-pam)
        |-sudo-+-su-+-bash
        |-sudo-+-su-+-bash
        |-systemd-udevd
        |-systemd-logind
        |-cpptools-srv
        |-cpptools-srv
```

4. pstree -V

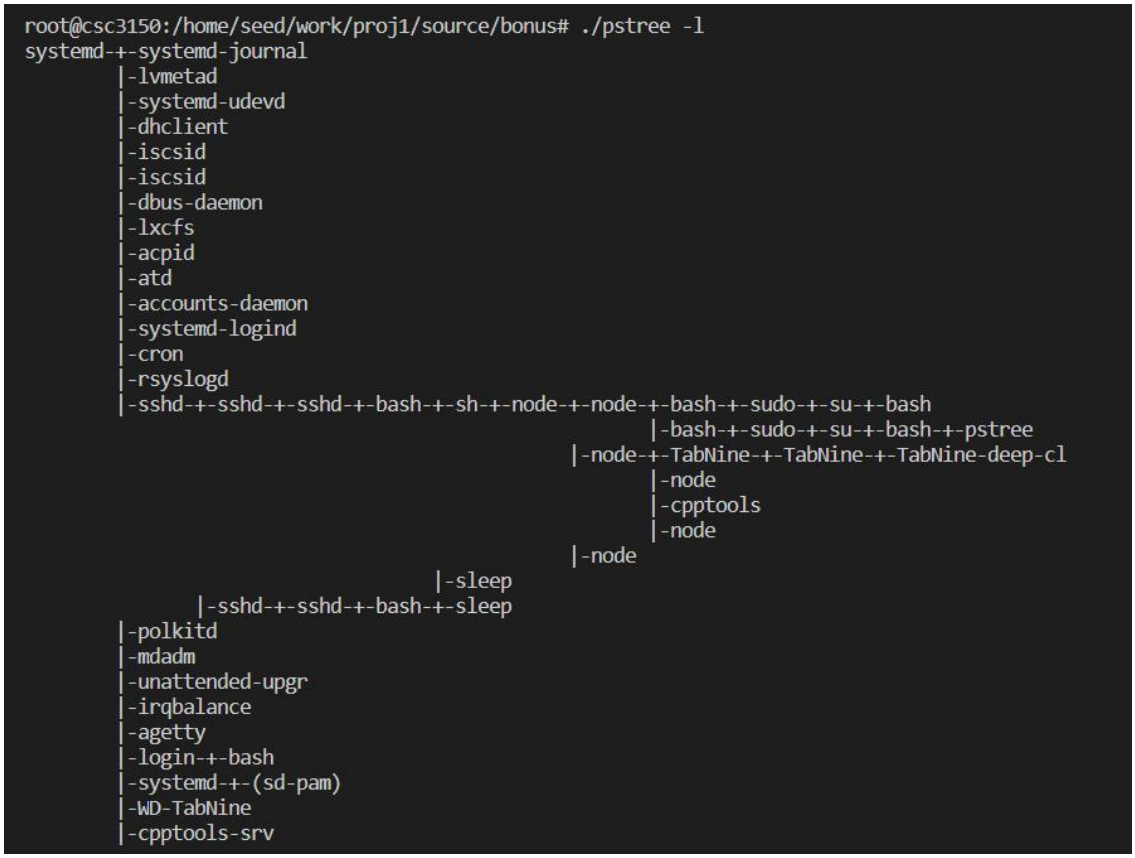
```
root@csc3150:/home/seed/work/proj1/source/bonus# ./pstree -V
pstree (PSmisc) 22.21
Copyright (C) 1993-2009 Werner Almesberger and Craig Small

PSmisc comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under
the terms of the GNU General Public License.
For more information about these matters, see the files named COPYING.
```

5. pstree -A

```
root@csc3150:/home/seed/work/proj1/source/bonus# ./pstree -A
systemd-+-systemd-journal
        |-lvmetad
        |-systemd-udevd
        |-dhclient
        |-iscsid
        |-iscsid
        |-dbus-daemon
        |-lxcfs
        |-acpid
        |-atd
        |-accounts-daemon
        |-systemd-logind
        |-cron
        |-rsyslogd
        |-sshd-+-sshd-+-sshd-+-bash-+-sh-+-node-+-node-+-bash-+-sudo-+-su-+-bash
                                                              |-bash-+-sudo-+-su-+-bash-+-pstree
                                                    |-node-+-TabNine-+-TabNine-+-TabNine-deep-cl
                                                              |-node
                                                              |-cpptools
                                                              |-node
                                                    |-node
                                          |-sleep
                |-sshd-+-sshd-+-bash-+-sleep
        |-polkitd
        |-mdadm
        |-unattended-upgr
        |-irqbalance
        |-agetty
        |-login-+-bash
        |-systemd-+-(sd-pam)
        |-WD-TabNine
        |-cpptools-srv
```

6. pstree -l



PART V About what I've learnt

In this project, I've learnt to fork a child process and execute the task both in user mode and kernel mode. More importantly, I gained a basic understanding of Linux source code. In the process of out exploration, my friends at Piazza and I were the first to solve the problem of do_wait() returning -10 and do_execve() returning -14 by digging into the Linux kernel source code. In a way, this is a contribution to the update of the project.

PartVI How to run my program

In your command line, type and enter:

- Program1:
    1. cd ./program1
    2. make
    3. ./program1 ./filename
- Program2:
    1. cd ./program2
    2. gcc test.c -o test
    3. make
    4. insmod program2.ko
    5. rmmod program2.ko (if the test would do abort(n), be sure to enter this line after n seconds)
    6. dmesg | tail -n 10
- Bonus:
    1. cd ./bonus
    2. make
    3. ./pstree (-n, -V, -p, -A, -l or just no parameter)

(You might need to export functions in linux kernel and recompile first)