Report of assignment 1 (121020150 Mang Qiuyang)

Task 1

environment:

Setup my `VM BOX` via the guidance on BB, and no obstacles met.

Change my Linux resources to the `Tsing-hua` resource.

Install the clang-format tools by `sudo apt-get install clang-format`.

my formulation:

Use `fork()` to create `child process`.

Use `pid` to check whether `parent process` or `child process`.

Use `execve(argv[1], argv, NULL)` to execute the child process, `envp=NULL` to ensure execute successfully.

Use `waitpid(pid, &status, WUNTRACED)` to wait the child process and catch the status of the child status, and using `WUNTRACED` instead of other options is in order to solve the case of `stop` .

Use `WIFEXITED`, `WIFSIGNALED`, `WIFSTOPPED`, et al. to receive the signal of the child process.

my outputs:

abort

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 2260
I'm the Child Process, my pid = 2261
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
child process get SIGABRT signal
CHILD EXCECUTION FAILED: 6
```

alarm

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./alarm
I'm the Child Process, my pid = 2279
Child process start to execute test program:
Process start to fork
I'm the Parent Process, my pid = 2278
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
CHILD EXCECUTION FAILED: 14
```

bus

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 2323
I'm the Child Process, my pid = 2324
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
CHILD EXCECUTION FAILED: 7
```

floating

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 2366
I'm the Child Process, my pid = 2367
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
CHILD EXCECUTION FAILED: 8
```

hangup

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 2395
I'm the Child Process, my pid = 2396
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
CHILD EXCECUTION FAILED: 1
```

illegal_instr

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 2435
I'm the Child Process, my pid = 2436
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process get SIGILL signal
CHILD EXCECUTION FAILED: 4
```

interrupt

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 2452
I'm the Child Process, my pid = 2453
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
CHILD EXCECUTION FAILED: 2
```

kill

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 2480
I'm the Child Process, my pid = 2481
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
CHILD EXCECUTION FAILED: 9
```

normal

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 2509
I'm the Child Process, my pid = 2510
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
```

pipe

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 2538
I'm the Child Process, my pid = 2539
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
CHILD EXCECUTION FAILED: 13
```

quit

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 2567
I'm the Child Process, my pid = 2568
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
CHILD EXCECUTION FAILED: 3
```

segment_fault

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 2602
I'm the Child Process, my pid = 2603
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
CHILD EXCECUTION FAILED: 11
```

stop

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 2631
I'm the Child Process, my pid = 2632
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
child process get SIGSTOP signal
CHILD PROCESS STOPPED: 19
```

terminate

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 2661
I'm the Child Process, my pid = 2662
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
CHILD EXCECUTION FAILED: 15
```

trap

```
vagrant@csc3150:~/csc3150/assignment1/source/program1$ ./program1 ./trap
Process start to fork
I'm the Child Process, my pid = 2678
Child process start to execute test program:
I'm the Parent Process, my pid = 2677
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
child process get SIGTRAP signal
CHILD EXCECUTION FAILED: 5
```

what I learned:

understand the basic concept of the process and the thread and their difference.

understand the process of the fork.

learn how to use the function `fork`, `execve`, `wait` to create, wait, and receive the child process.

Task 2

environment:

Download the kernel with `5.10.x` version from the website.

Unzip it and export the functions will be used from the head files.

`kernel_clone` from the `kernel/fork.c`

`do_wait` from the `kernel/exit.c`

`getname_kernel` from the `fs/namei.c`

`do_execve` from the `fs/exec.c`

Then compile my kernel via the guidance on the tut 2.

Problems met and corresponding solutions during this process:

The memory distributed to my `VM box` is too small: redistribute it to the 4096 MB.

can not open the `menuconfig`: drag my termination to the enough big size.

Finally use the `insmod`, `rmmod` and `dmesg` to test my program.

formulation:

First, `extern` all needed functions:

```
extern pid_t kernel_clone(struct kernel_clone_args *kargs);
extern int do_execve(struct filename *filename,
                     const char __user *const __user *__argv,
                     const char __user *const __user *__envp);
extern struct filename *getname_kernel(const char *filename);
struct wait_opts {
  enum pid_type wo_type;
  int wo_flags;
  struct pid *wo_pid;
  struct waitid_info *wo_info;
  int wo_stat;
  struct rusage *wo_rusage;
  wait_queue_entry_t child_wait;
  int notask_error;
};
extern long do_wait(struct wait_opts *wo);
```

Use `kthread_create` and `wake_up_process` to fork my process.

Use `kernel_clone` to implement the similar function of `fork` to invoke `my_exec`

```
struct kernel_clone_args clone_args = {
    .flags = SIGCHLD,
    .child_tid = NULL,
    .parent_tid = NULL,
    .exit_signal = SIGCHLD,
    .stack_size = 0,
    .tls = 0,
    .stack = (unsigned long)&my_exec,
};
```

The return value of the `kernel_clone` is the `pid` of the `child process`, and use the `task_pid_nr(current)` to get the `pid` of the `parent process`.

Use `getname_kernel` to get the filename of the test file, the path of the `getname_kernel` must be the **absolute path**.

Use `do_execve` to execute the test file and the `args = envp = NULL` to ensure execute successfully.

```
int return_value = do_execve(
    getname_kernel("/home/vagrant/csc3150/assignment1/source/program2/test"),
    NULL, NULL);
```

Check whether the child process execute successfully by checking the return value (return_value = 0 means OK).

The part of `do_wait` is a bit complicated:

Define a C: struct `wait_opts` from the `kernel/exit.c`

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

Notice: that `wo_stat` should be a int not a pointer.

Initialize my `wo` and use `wo` to wait the child process:

```
struct pid *wo_pid = NULL;
enum pid_type 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_stat = 0;
wo.wo_rusage = NULL;
int return_value = do_wait(&wo);
```

Notice: `wo.wo_flags` should be `WEXITED | WUNTRACED` to fit the all test case .

Then we need solve the different type of caught signal like `Task1`, but we can not invoke functions like `WIFEXITED`, `WIFSIGNALED`, `WIFSTOPPED`. **However I read the `sys/wait` and write my version of these function**:

```
#define my_WAIT_INT(status) (*(__const int *)&(status))
#define my_WEXITSTATUS(status) (((status)&0xff00) >> 8)
#define my_WTERMSIG(status) ((status)&0x7f)
#define my_WSTOPSIG(status) my_WEXITSTATUS(status)
#define my_WIFEXITED(status) (my_WTERMSIG(status) == 0)
#define my_WIFSIGNALED(status) (((signed char)(((status)&0x7f) + 1) >> 1) > 0)
#define my_WIFSTOPPED(status) (((status)&0xff) == 0x7f)
#define WEXITSTATUS(status) my_WEXITSTATUS(my_WAIT_INT(status))
#define WTERMSIG(status) my_WTERMSIG(my_WAIT_INT(status))
#define WSTOPSIG(status) my_WSTOPSIG(my_WAIT_INT(status))
#define WIFEXITED(status) my_WIFEXITED(my_WAIT_INT(status))
#define WIFSIGNALED(status) my_WIFSIGNALED(my_WAIT_INT(status))
#define WIFSTOPPED(status) my_WIFSTOPPED(my_WAIT_INT(status))
```

Combined with the `wait` part of `Task 1`, the program has received all of 15 types signal successfully.

my output

alarm

```
root@csc3150:~/csc3150/assignment1/source/program2# dmesg
[  666.681459] [program2] : Module_exit
[  668.090790] [program2] : Module_init {Mang Qiuyang} {121020150}
[  668.090907] [program2] : module_init create kthread start
[  668.090933] [program2] : module_init kthread start
[  668.090948] [program2] : The child process has pid = 3259
[  668.090949] [program2] : This is the parent process, pid = 3258
[  668.090952] [program2] : child process
[  670.096852] [program2] : get SIGALRM signal
[  670.096855] [program2] : child process terminated
[  670.096857] [program2] : The return signal is 14
```

abort

```
[  740.743850] [program2] : Module_exit
[  743.449535] [program2] : Module_init {Mang Qiuyang} {121020150}
[  743.449595] [program2] : module_init create kthread start
[  743.450831] [program2] : module_init kthread start
[  743.450849] [program2] : The child process has pid = 3422
[  743.450850] [program2] : This is the parent process, pid = 3420
[  743.450853] [program2] : child process
[  743.621811] [program2] : get SIGABRT signal
[  743.621814] [program2] : child process terminated
[  743.621815] [program2] : The return signal is 6
```

bus

```
[  775.470759] [program2] : Module_exit
[  776.602695] [program2] : Module_init {Mang Qiuyang} {121020150}
[  776.602853] [program2] : module_init create kthread start
[  776.602893] [program2] : module_init kthread start
[  776.602909] [program2] : The child process has pid = 3487
[  776.602909] [program2] : This is the parent process, pid = 3486
[  776.602912] [program2] : child process
[  776.775658] [program2] : get SIGBUS signal
[  776.775660] [program2] : child process terminated
[  776.775661] [program2] : The return signal is 7
```

floating

```
[  812.094215] [program2] : Module_exit
[  813.198526] [program2] : Module_init {Mang Qiuyang} {121020150}
[  813.198637] [program2] : module_init create kthread start
[  813.199732] [program2] : module_init kthread start
[  813.199748] [program2] : The child process has pid = 3537
[  813.199749] [program2] : This is the parent process, pid = 3535
[  813.199751] [program2] : child process
[  813.373313] [program2] : get SIGFPE signal
[  813.373315] [program2] : child process terminated
[  813.373316] [program2] : The return signal is 8
```

hangup

```
[  852.518603] [program2] : Module_exit
[  853.844987] [program2] : Module_init {Mang Qiuyang} {121020150}
[  853.845366] [program2] : module_init create kthread start
[  853.846659] [program2] : module_init kthread start
[  853.846680] [program2] : The child process has pid = 3607
[  853.846680] [program2] : This is the parent process, pid = 3605
[  853.846683] [program2] : child process
[  853.847130] [program2] : get SIGHUP signal
[  853.847132] [program2] : child process terminated
[  853.847133] [program2] : The return signal is 1
```

illegal_instr

```
[  899.490931] [program2] : Module_exit
[  900.878533] [program2] : Module_init {Mang Qiuyang} {121020150}
[  900.878792] [program2] : module_init create kthread start
[  900.878837] [program2] : module_init kthread start
[  900.878864] [program2] : The child process has pid = 3670
[  900.878865] [program2] : This is the parent process, pid = 3669
[  900.878872] [program2] : child process
[  901.050119] [program2] : get SIGILL signal
[  901.050122] [program2] : child process terminated
[  901.050123] [program2] : The return signal is 4
```

interrupt

```
[  938.966968] [program2] : Module_exit
[  940.310009] [program2] : Module_init {Mang Qiuyang} {121020150}
[  940.310133] [program2] : module_init create kthread start
[  940.310398] [program2] : module_init kthread start
[  940.310411] [program2] : The child process has pid = 3753
[  940.310412] [program2] : This is the parent process, pid = 3752
[  940.310414] [program2] : child process
[  940.310791] [program2] : get SIGINT signal
[  940.310791] [program2] : child process terminated
[  940.310792] [program2] : The return signal is 2
```

kill

```
[  975.589316] [program2] : Module_exit
[  976.772733] [program2] : Module_init {Mang Qiuyang} {121020150}
[  976.772768] [program2] : module_init create kthread start
[  976.772921] [program2] : module_init kthread start
[  976.772930] [program2] : The child process has pid = 3820
[  976.772930] [program2] : This is the parent process, pid = 3819
[  976.772933] [program2] : child process
[  976.773313] [program2] : get SIGKILL signal
[  976.773314] [program2] : child process terminated
[  976.773315] [program2] : The return signal is 9
```

normal

```
[ 1004.190588] [program2] : Module_exit
[ 1005.526093] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1005.526241] [program2] : module_init create kthread start
[ 1005.526385] [program2] : module_init kthread start
[ 1005.526398] [program2] : The child process has pid = 3866
[ 1005.526398] [program2] : This is the parent process, pid = 3865
[ 1005.526727] [program2] : child process
[ 1005.527209] [program2] : child process terminated
[ 1005.527211] [program2] : The return signal is 0
```

pipe

```
[ 1040.413582] [program2] : Module_exit
[ 1041.593825] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1041.593860] [program2] : module_init create kthread start
[ 1041.594917] [program2] : module_init kthread start
[ 1041.594936] [program2] : The child process has pid = 3914
[ 1041.594936] [program2] : This is the parent process, pid = 3912
[ 1041.594939] [program2] : child process
[ 1041.595294] [program2] : get SIGPIPE signal
[ 1041.595295] [program2] : child process terminated
[ 1041.595296] [program2] : The return signal is 13 _
```

quit

```
[ 1087.489171] [program2] : Module_exit
[ 1088.877653] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1088.877684] [program2] : module_init create kthread start
[ 1088.877819] [program2] : module_init kthread start
[ 1088.877830] [program2] : The child process has pid = 3994
[ 1088.877831] [program2] : This is the parent process, pid = 3993
[ 1088.877833] [program2] : child process
[ 1089.054113] [program2] : get SIGQUIT signal
[ 1089.054116] [program2] : child process terminated
[ 1089.054118] [program2] : The return signal is 3
```

segment_fault

```
[ 1142.585766] [program2] : Module_exit
[ 1144.091036] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1144.091199] [program2] : module_init create kthread start
[ 1144.092114] [program2] : module_init kthread start
[ 1144.092137] [program2] : The child process has pid = 4118
[ 1144.092138] [program2] : This is the parent process, pid = 4116
[ 1144.092504] [program2] : child process
[ 1144.268154] [program2] : get SIGSEGV signal
[ 1144.268158] [program2] : child process terminated
[ 1144.268160] [program2] : The return signal is 11 _
```

stop

```
[ 1174.588900] [program2] : Module_exit
[ 1175.837128] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1175.837173] [program2] : module_init create kthread start
[ 1175.837304] [program2] : module_init kthread start
[ 1175.837316] [program2] : The child process has pid = 4186
[ 1175.837317] [program2] : This is the parent process, pid = 4185
[ 1175.838205] [program2] : child process
[ 1175.839175] [program2] : get SIGSTOP signal
[ 1175.839177] [program2] : child process terminated
[ 1175.839179] [program2] : The return signal is 19 _
```

terminate

```
[ 1210.810336] [program2] : Module_exit
[ 1212.734734] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1212.734860] [program2] : module_init create kthread start
[ 1212.734886] [program2] : module_init kthread start
[ 1212.734907] [program2] : The child process has pid = 4281
[ 1212.734908] [program2] : This is the parent process, pid = 4280
[ 1212.734910] [program2] : child process
[ 1212.735306] [program2] : get SIGTERM signal
[ 1212.735307] [program2] : child process terminated
[ 1212.735308] [program2] : The return signal is 15
```

trap

```
[ 1247.769235] [program2] : Module_exit
[ 1249.199653] [program2] : Module_init {Mang Qiuyang} {121020150}
[ 1249.199682] [program2] : module_init create kthread start
[ 1249.199806] [program2] : module_init kthread start
[ 1249.199815] [program2] : The child process has pid = 4328
[ 1249.199816] [program2] : This is the parent process, pid = 4327
[ 1249.199818] [program2] : child process
[ 1249.372717] [program2] : get SIGTRAP signal
[ 1249.372719] [program2] : child process terminated
[ 1249.372720] [program2] : The return signal is 5
```

what I learned

Read a lot of the underlying code related to the kernel.

Understand the basic implement of the functions: `do_wait`, `fork`.

Get deeper perspective of the process and thread.

Learned some kernel programming.

Learned how to compile kernel and insert/remove my kernel.