

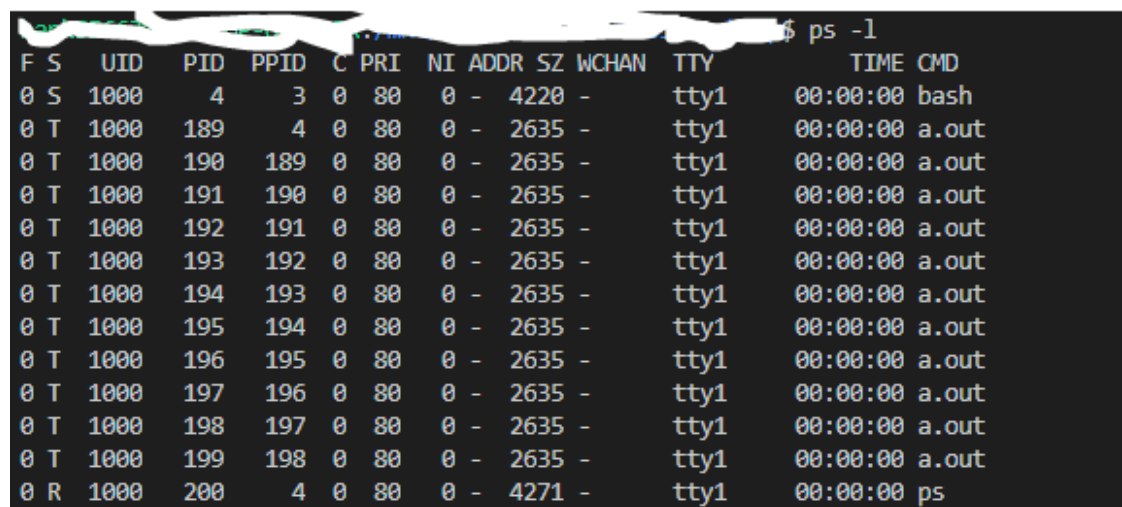
Project #1. Process creation, cleanup, and IPC

Project submission guidelines

1. Use Linux for this project, do not use Cygwin.
2. For program code, use C/C++ and name your source file as taskX.cpp or taskX.c, where X is the task number.
3. For questions & answers, write your answers in a single pdf file and name the file as report.pdf. You can answer in Chinese or English.
4. Store all the source files and the report.pdf in a directory. Use your student ID as the directory name. Compress the directory into a .zip named by your student ID such as 07XXXXX.zip.

Process creation and cleanup

Under Linux, we can use the command “ps -l” to observe the process running on the system.



F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	1000	4	3	0	80	0	-	4220	-	tty1	00:00:00	bash
0	T	1000	189	4	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	190	189	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	191	190	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	192	191	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	193	192	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	194	193	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	195	194	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	196	195	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	197	196	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	198	197	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	T	1000	199	198	0	80	0	-	2635	-	tty1	00:00:00	a.out
0	R	1000	200	4	0	80	0	-	4271	-	tty1	00:00:00	ps

PID : each process's ID ◦

PPID : each process's parent ID ◦

PGID : (not shown above) each process's process group ID. You can use "ps xao pgid,comm" to show the PGID information ◦

● Task 2 (program code) [30%]:

Create a process tree consisting of 10 processes at 10 different levels. In other words, please have process P_1 as the parent process of P_2 , process P_2 as the parent process of P_3 , etc. Once the tree has been created, your program should start to terminate the processes. For the termination, you should terminate a process only after all its descendent processes have been terminated. In your program, please print the events of process creation and process termination (along with the pid information) on the screen.

```

process pid 1251 create
process pid 1252 create
process pid 1253 create
process pid 1254 create
process pid 1255 create
process pid 1256 create
process pid 1257 create
process pid 1258 create
process pid 1259 create
process pid 1260 create
process pid 1261 create
process pid 1261 exit
process pid 1260 exit
process pid 1259 exit
process pid 1258 exit
process pid 1257 exit
process pid 1256 exit
process pid 1255 exit
process pid 1254 exit
process pid 1253 exit
process pid 1252 exit
process pid 1251 exit

```

- **Task 3 (program code) [30%]**

Fix the code in `process_2.cpp` such that each process will only exit after all its descendent processes have exited. Also, print the parent-child relations (i.e., process X created process Y) on the screen as shown below.

```

create main process 1267
process 1267 create process 1268
process 1268 create process 0
process 1267 create process 1269
process 1269 create process 0
process 1268 create process 1270
process 1270 create process 0
process 1267 create process 1271
process 1271 create process 0
process 1271 exit its child process 0 0 0
process 1269 create process 1272
process 1272 create process 0
process 1272 exit its child process 0 0 0
process 1269 exit its child process 0 0 1272
process 1268 create process 1273
process 1273 create process 0
process 1273 exit its child process 0 0 0
process 1270 create process 1274
process 1274 create process 0
process 1274 exit its child process 0 0 0
process 1270 exit its child process 0 0 1274
process 1268 exit its child process 0 1270 1273
process 1267 exit its child process 1268 1269 1271

```

IPC

- **Task 5** (program code) [40%]:
 - Write a program to play paper-scissor-stone with TA's program. There are a total of 100 rounds of matches. In each round, if you win, you will get 1 point. At the start of each round, TA's program will output "OK" to the screen (standard output). After your program sees that, your program can write the choice of paper, scissor, or stone to the standard output. At the same time, TA's program will also output his choice to standard output. For convenience, we encode the choices as an integer, where 0 represents Paper, 1 represents Scissor, and 2 represents Stone.
 - To give your program a little advantage, TA's program will leak his choice to a shared memory prior to showing the "OK" message. The key of the shared memory will be provided in the beginning from TA's program, you can read it by standard input.
 - Print your choices to standard output (e.g., cout or printf). Right after you print a choice, use fflush(stdout) or cout.flush() to flush the buffer. If you still have doubts regarding the input/output format, please have a look at the output from TA's program.
 - Os_fake_sol.cpp is a sample program that plays with the TA's program by making random choices.
 - Os_fake_judge.cpp is a sample program that always chooses Paper, other action is same as TA's program. It will help you to complete your project. And please replace the parameter of function ftok to a valid pathname on your computer.

- `interactive.sh` is an interactive script ,First compile the file `Os_fake_judge.cpp` and your solution to executable file, and use the following command
`./interactive.sh ./{Os_fake_judge.cpp executable file} ./{your solution executable file}`, and it will show your final score. If it is not showing anything, that means your program is not functioning properly. You will get 0 points for this task in this case.