CSCI 4061 Recitation 3

Feb 7,2022

Contents

- fork() and wait()
- Exec()
- Breakout Activity

fork()

- System call to create a new process
- Child is a clone of parent process
- Parent and child runs in separate memory spaces
- At time of fork, both will have same content

fork()

- Syntax
  - pid_t fork(void);
- Return values
  - pid > 0 ➔ success, **Parent** execution
  - pid = 0 ➔ success, **Child** execution
  - pid = −1 ➔ failure, no child created

- Header
  - #include <sys/types.h>
  - #include <unistd.h>
  - #include <errno.h>

wait()

- Wait for state change in a child of the calling process
- State change: child terminated; child stopped by a signal; child resumed by a signal
- On termination, performing a wait allows system to reclaim resources
- If wait not added, terminated child will remain in zombie state as resources are not released

wait()

- Header
  - #include <sys/types.h>
  - #include <sys/wait.h>

- Syntax
  - pid_t wait(int *wstatus);
  - pid_t waitpid(pid_t pid, int *wstatus, int options);
- Return values
  - wait(): Returns process ID of terminated child on success, else returns -1 and errors
  - waitpid(): Returns process ID of child on state change, else return -1 and errors

exec()

- Deletes the current program state and begins executing a new program.
- If successful, will not return to old program
- Has several variants: execl, execv, execp, execle

Exec Contains Character ...	Meaning
l	Consumes list of args of constant size. (ends with char* NULL)
v	Consumes array of args of variable size.
p	Consumes filename instead of path. Uses default OS ‘path’ to find file.
e	Overrides default environment (another way to pass args).

Code examples

Exercise

Problem description:

- Modify the given code to accept a numeric value *n* via command line
- The program should then create exactly *n* child processes
- Each child will print their process id and call “execl” to print out “hello there”
- Once all the child processes exit, the parent will create one more child
- This child will execute the given “ptime” executable using “execv” and terminate. There is no input argument required for “ptime”.

Type “chmod +x ptime” in the terminal, if you get a “permission denied” error

---

Submission Instructions: Submit only main.c file **DO NOT SUBMIT A TAR FILE**

Exercise output

```
$ gcc -o main main.c
$ ./main 5
42104
42105
42106
42107
42108
hello there
hello there
hello there
hello there
hello there
Present time: Sat Feb 5 11:51:48 2022
```

The process id can be different for your output. Also the print statements may not be ordered.

Individual Submission

- Submit main.c file to Canvas