Project 1: NPShell

NP TA 孟宇

10/20 18:20

Project 1 Deadline

Demo: 10/22 Thu.

General Info

- We will announce our Bitbucket organization.
- We will announce the nplinux account.
  - NP projects should run on NP servers.
  - Any abuse of NP server will be recorded.
  - Don't leave any zombie processes in the system.

Project 1: Info

- You are HIGHLY encouraged to publish your questions on Project 1 討論區
- You can contact TAs by email: np@cgilab.nctu.edu.tw (Mails sent to other addresses will NOT be replied)
- TA hours (Thursday: 15:00 ~ 17:00) on 10/8, 10/15 will be held at EC511
- TAs will NOT debug for you.

Project 1: Submission

- Create a directory named as your student ID, put all files into the directory.
- You **MUST** use GNU Make to build your project and compile your source code into one executable named npshell. The executable and Makefile should be placed at the top layer of the directory. We will use this executable for demo.
- You are NOT allowed to demo if we are unable to compile your project with a single make command.
- Upload only your code and Makefile. Do NOT upload anything else (e.g. noop, removetag, test.html, **.git**, **__MACOSX**)
- zip the directory and upload the .zip file to the E3 platform.
  ATTENTION! We only accept .zip format

4. **zip** the directory and upload the .zip file to the E3 platform

   ATTENTION! We only accept .zip format

   e.g.

   Create a directory 0856053, the directory structure may be:

```
0856053
├── Makefile
├── shell.cpp
└── shell.h
```

   zip the folder 0856053 into 0856053.zip and upload 0856053.zip onto E3

G. We take plagiarism seriously.

All projects will be checked by a cutting-edge plagiarism detector.

You will get zero points on this project for plagiarism.

Please don't copy-paste any code from the internet, this may be considered plagiarism as well.

Protect your code from being stolen.

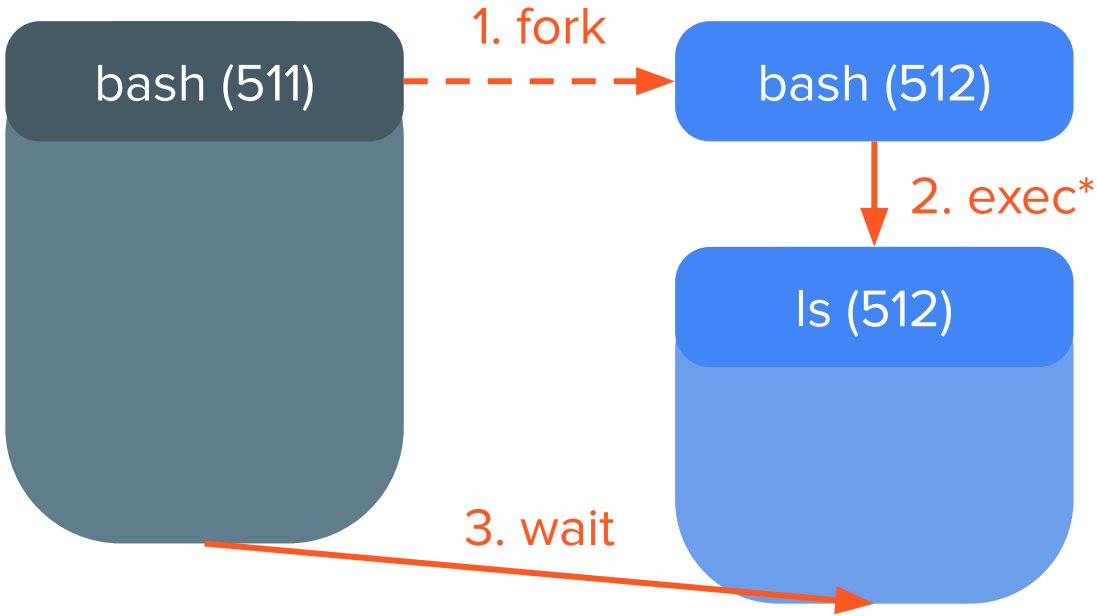
Project 1: Demo

- 10/22 Thu. 13:10 ~ 21:30
- We will announce demo slots 2~3 days before.
- Tasks:
  - (correct format and compile)
  - QA?
  - Pass np-basic test cases
  - Pass np-hard test cases
  - Implement 1 or 2 extra functions with limited time

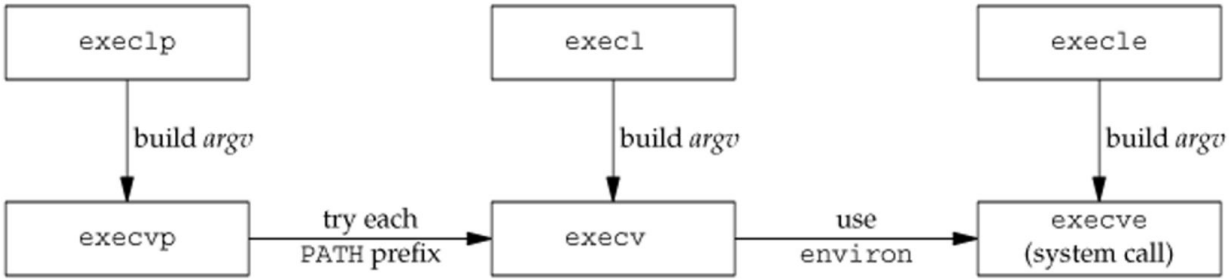
Implementation

Process Lifecycle: fork-exec-wait

1. Creation: fork
2. Execution exec*
3. Termination: waitpid

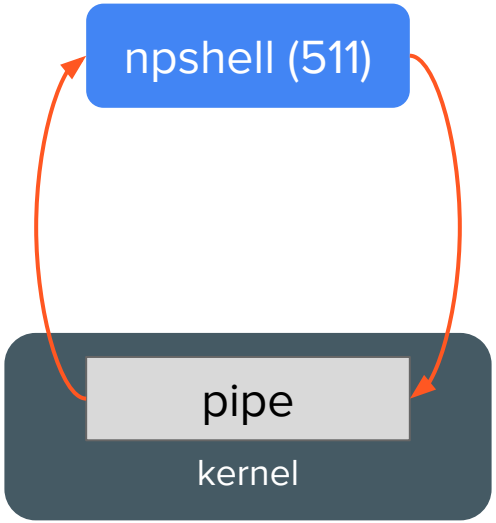


exec*

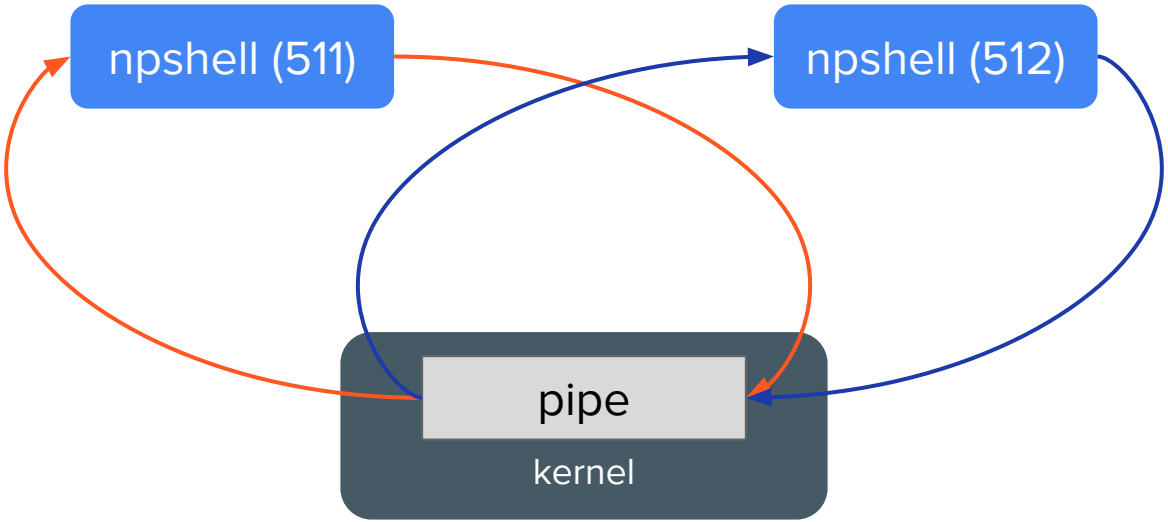

execlp
execl
execle
build argv
build argv
build argv
execvp
try each
PATH prefix
execv
use
environ
execve
(system call)

Pipe

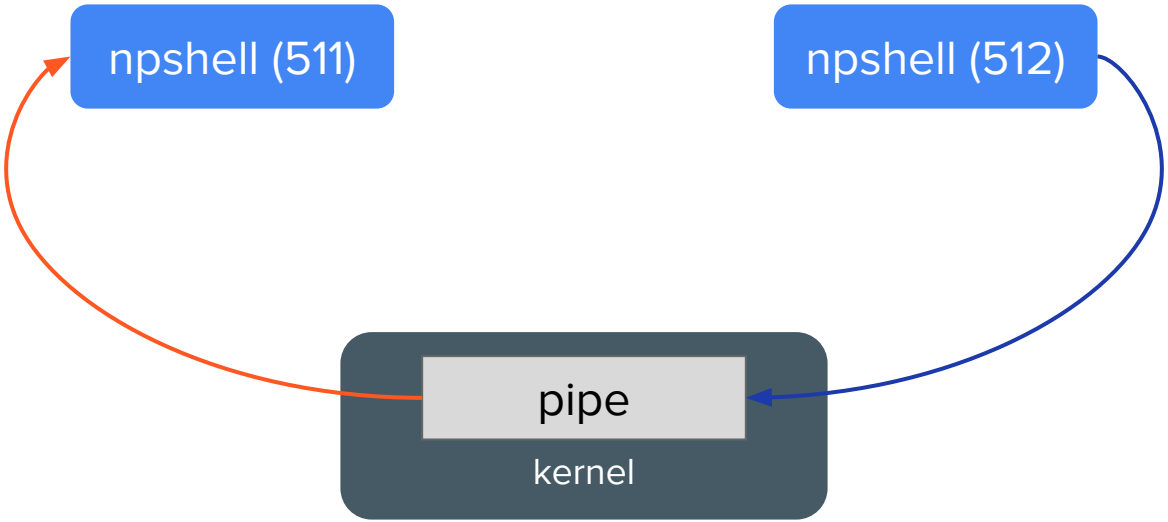
0. pipe



1. fork

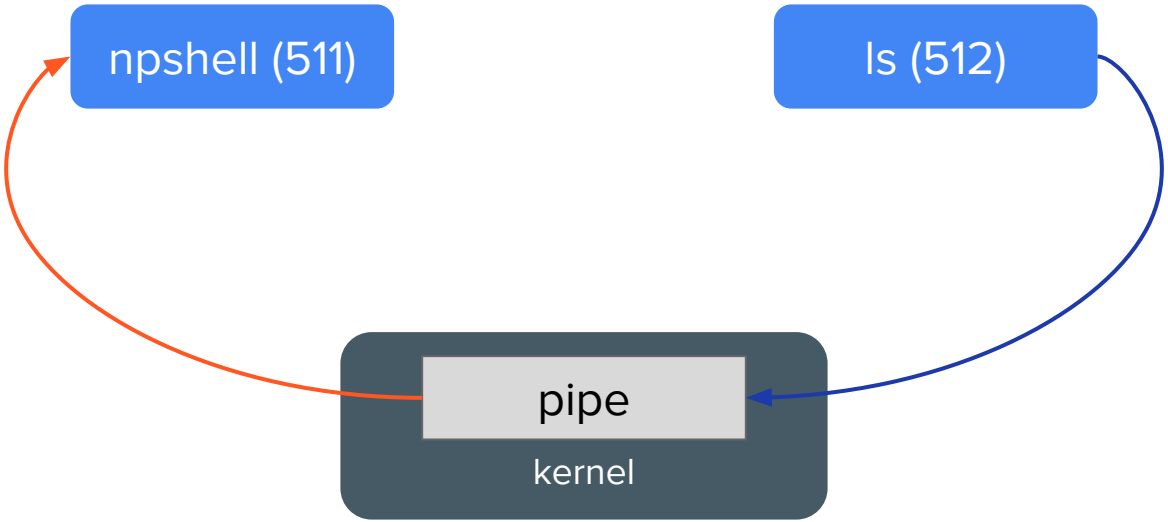


2. close



(because pipe is half-duplex)

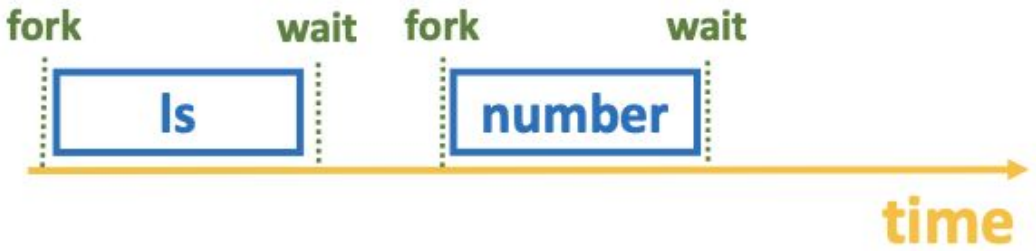
3. exec



Issues

Impl 1 : Wait for each child

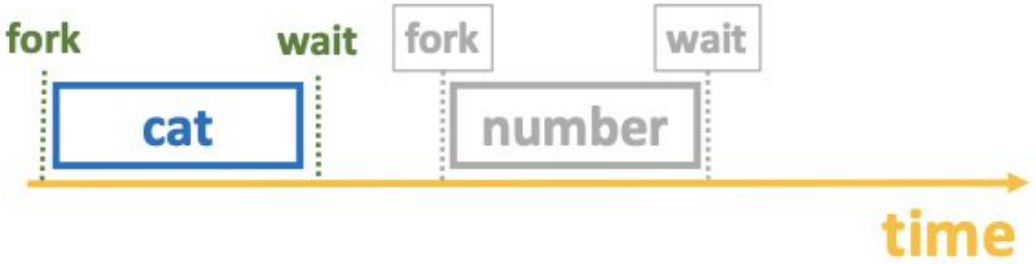
% ls | number



Problem 1 : Unable to process large data

```
% cat largeFile.txt | number
```

```
% cat largeFile.txt |1
% number
```



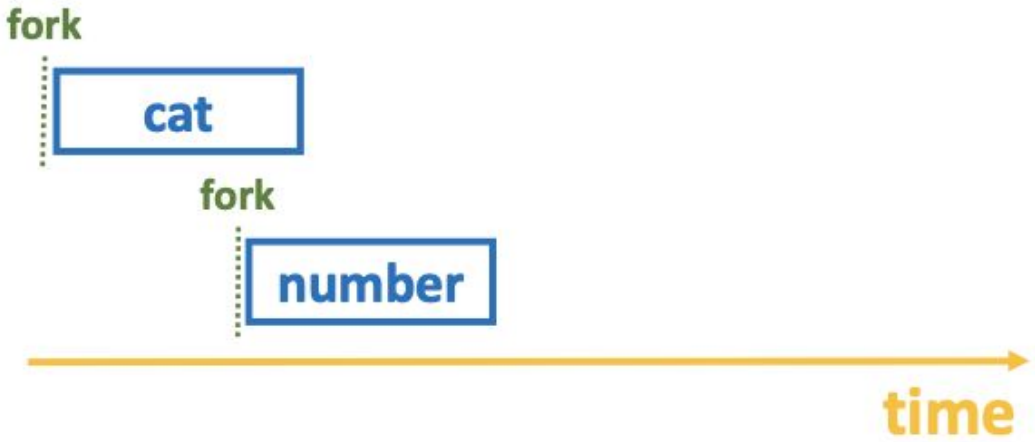
The process will hang forever !

Notes

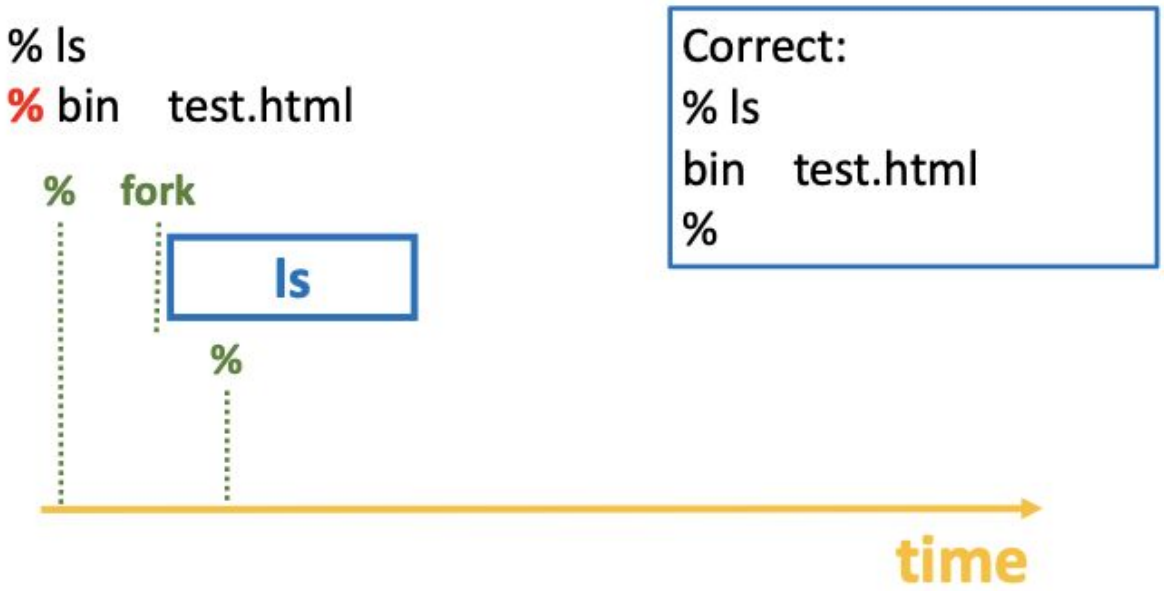
- Please don't store data into temporary files.

Impl 2 : Don't wait for processes

% cat largeFile.txt | number



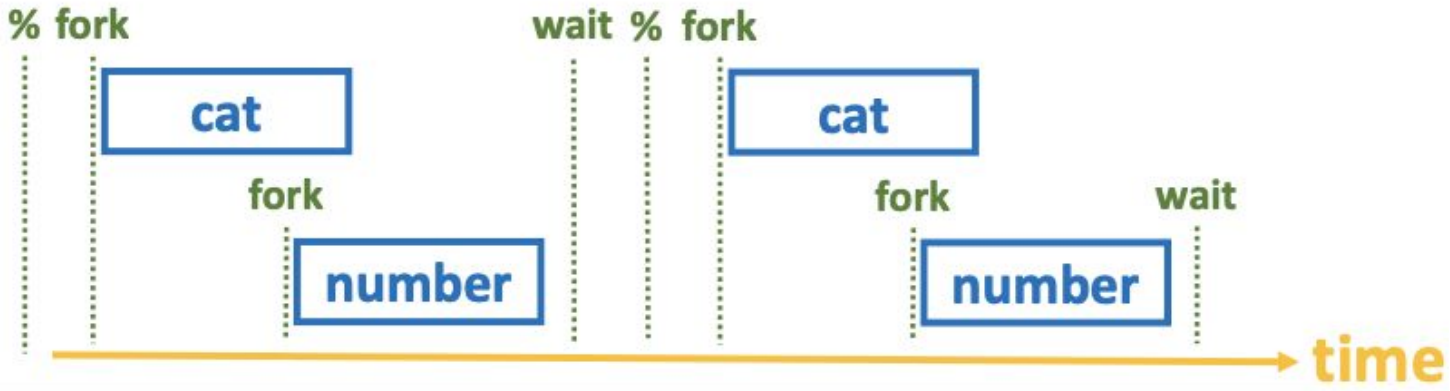
Problem 2 : % ordering



Correct:
% ls
bin test.html
%

Impl 3 : Wait for a line if not pipeN

```
% cat largeFile.txt | number
% cat largeFile.txt |1
% number
```

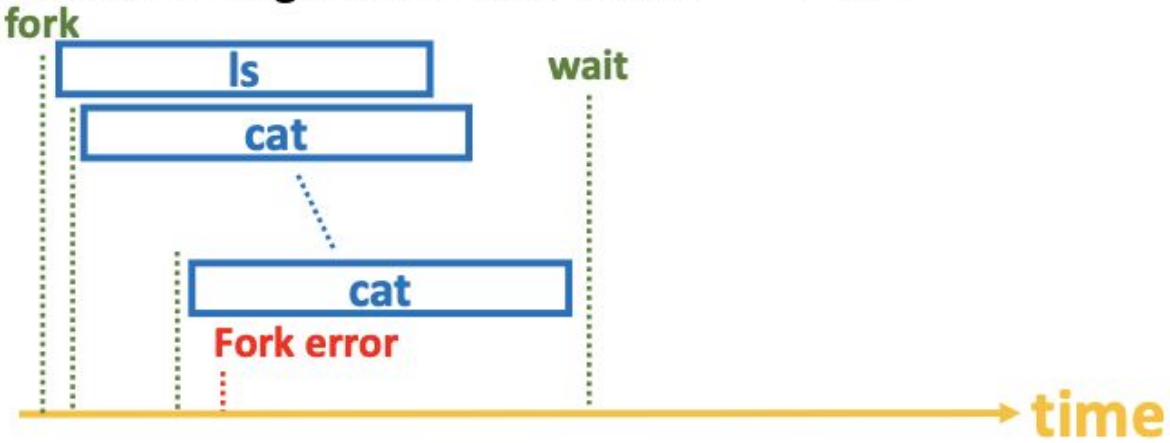


Problem 4 : Process limitation

ls | cat | cat | cat | cat ..... cat | cat

Process limit is **512** on NP server.

Fork error might occur around the **510**th cat.



More Problems...

不在此爆雷

Hint

- Functions you may use
  - fork
  - pipe
  - dup, dup2
  - exec*
  - wait, waitpid
- Handle failure
  - See man page for more information
- Debug
  - gdb
  - lsof

Enjoy the project!

Project 1 討論區

- 有空多逛逛討論區
- 鼓勵同學在討論區發問
- 私人問題再寄信給助教 (np@cgilab.nctu.edu.tw)

Project 1 額外提醒

- **Demo sample release**
  - Please run the demo script before demo
- **About bitbucket**
  - Commit your code >= 5 times on bitbucket
  - Repo name <student id>_np_project1
- **Lex & Yacc are NOT allowed**
- **C++ version ?**
  - g++ version on nplinux is 8.3.1
  - 以 nplinux **可以編譯執行為準**

Project 1 - About Signal

- **(Recommendation) Do not handle SIGINT**
  - (Recommendation) npshell should be killed by Ctrl-C
- **(Recommendation) Do not double fork**
  - Child return value?
- It is possible to finish this project **without** signal handler
  - But you need to understand the concept of process, **SIGCHLD**, fork, child, and **zombie**
  - Use wait / waitpid is enough

Project 1 問題 - getenv and setenv

- **system() function is NOT allowed**
- **The following function are allowed**
  - char *getenv(const char *name);
  - int setenv(const char *name, const char *value, int overwrite);
  - Think about if you can not you these functions
- **Do not manage environment variables by yourself.**

Project 1 問題 - 輸出順序

```
% removetag0 test.html |1
Error: illegal tag “!test.html”
%
```

```
% removetag0 test.html |1
% Error: illegal tag “!test.html”
```

Which one is correct ?

Project 1 問題 - 輸出順序

- **What is the output of % a | b | c | d ?**

  Unknown command: [a].

  Unknown command: [b].

  Unknown command: [c].

  Unknown command: [d].

Project 1 問題 - 輸出順序

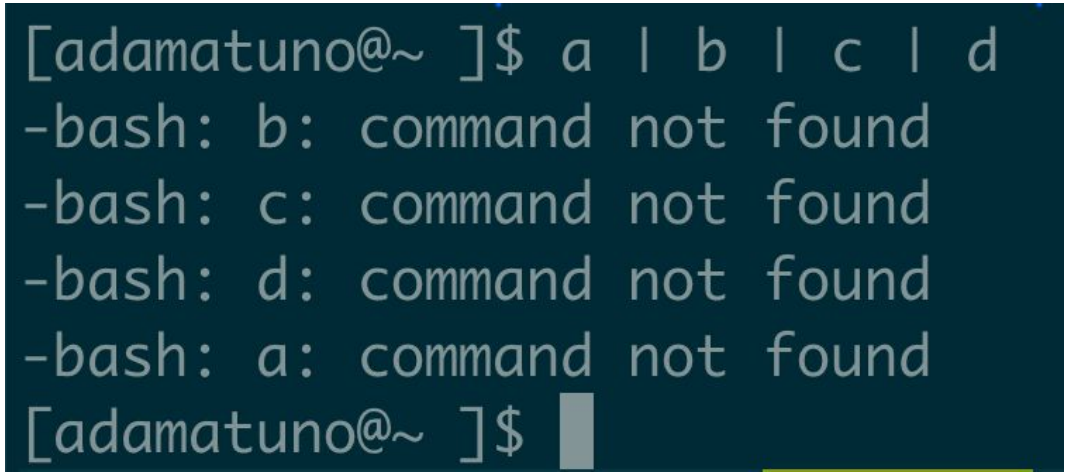
- **What is the output of % a | b | c | d ?**

Unknown command: [a].

Unknown command: [b].

Unknown command: [c].

Unknown command: [d].



Project 1 問題 - pipe 寫入順序

- What is the output of the following commands ?

  % cat many0.txt |2

  % cat many1.txt |1

  % cat

Project 1 問題 - 輸出順序

- 我們的測資會確保答案唯一，e.g.

  % cat many0.txt |2

  % cat many1.txt |1

  % wc -c

Project 1 問題 - Unknown command

- **Unknown command 視為只印出 stderr 然後馬上結束的程式。**

% ba > test.txt　　# 會建 test.txt

Unknown command: [ba].

- **Example**

% ba !1

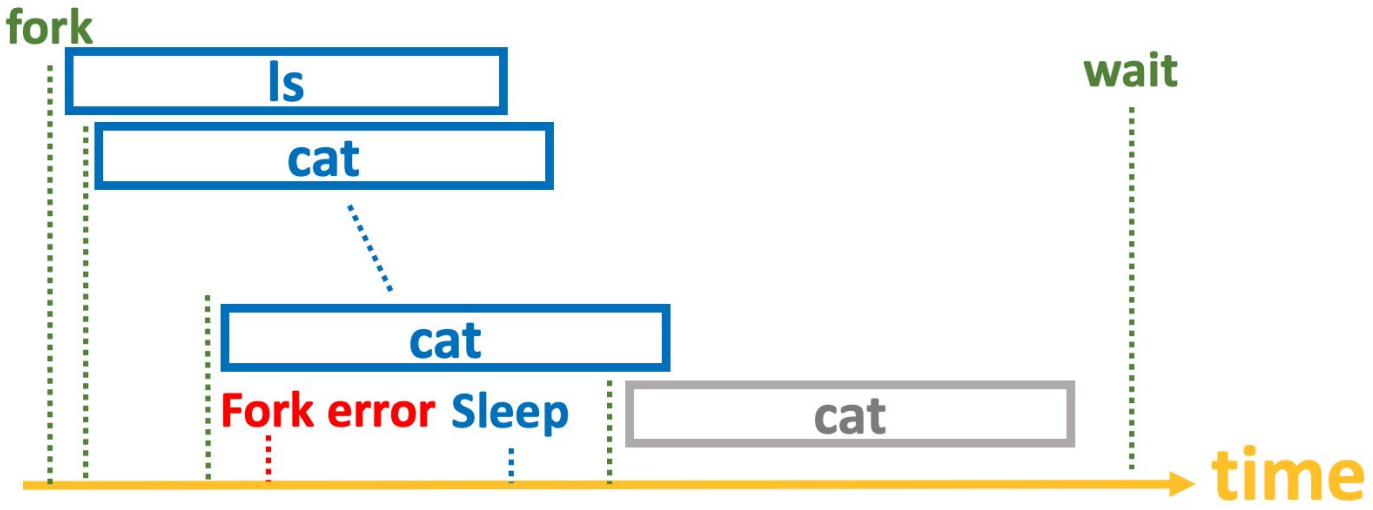
% cat

Unknown command: [ba].

Problem 5 : If fork error, sleep a little

ls | cat | cat | cat | cat ..... cat | cat

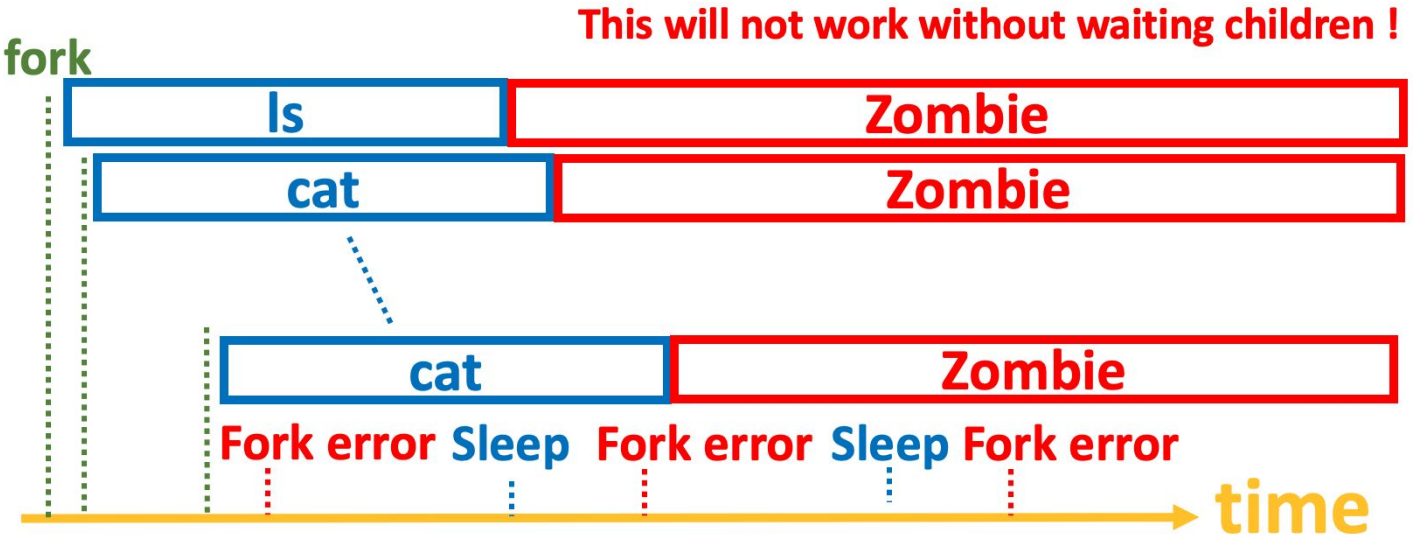
```
while ((pid = fork()) < 0) {
        usleep (1000);
}
```



Problem 5 : If fork error, sleep a little

ls | cat | cat | cat | cat ..... cat | cat

```
while ((pid = fork()) < 0) {
        usleep (1000);
}
```



Project 1 補充 - wait / waitpid

- **pid_t wait(int *wstatus);**
  - Suspends execution until one of its children terminates.
  - = waitpid(-1, &wstatus, 0);
- **pid_t waitpid(pid_t pid, int *wstatus, int options);**
  - Suspends execution until child **pid** terminates.
  - **waitpid(-1, &wstatus, WNOHANG)**
    - Return immediately if no child has exited.
- A child that terminates, but has not been waited for becomes a "**zombie**".
- See man page for more information

Implementation 6 : waitpid

ls | cat | cat | cat | cat ..... cat | cat

```
……
while (fork() < 0) {
      waitpid(-1, &status, 0);
}

……

If ( ! lineEndsWithPipeN) {
      for (pid in pidTableForCurrentLine) {
            int status;
            waitpid(pid, &status, 0);
      }
}
```

```
// Periodic call
waitpid(-1, &status, WNOHANG);
```

Implementation 7 : Signal Handler

ls | cat | cat | cat | cat ..... cat | cat

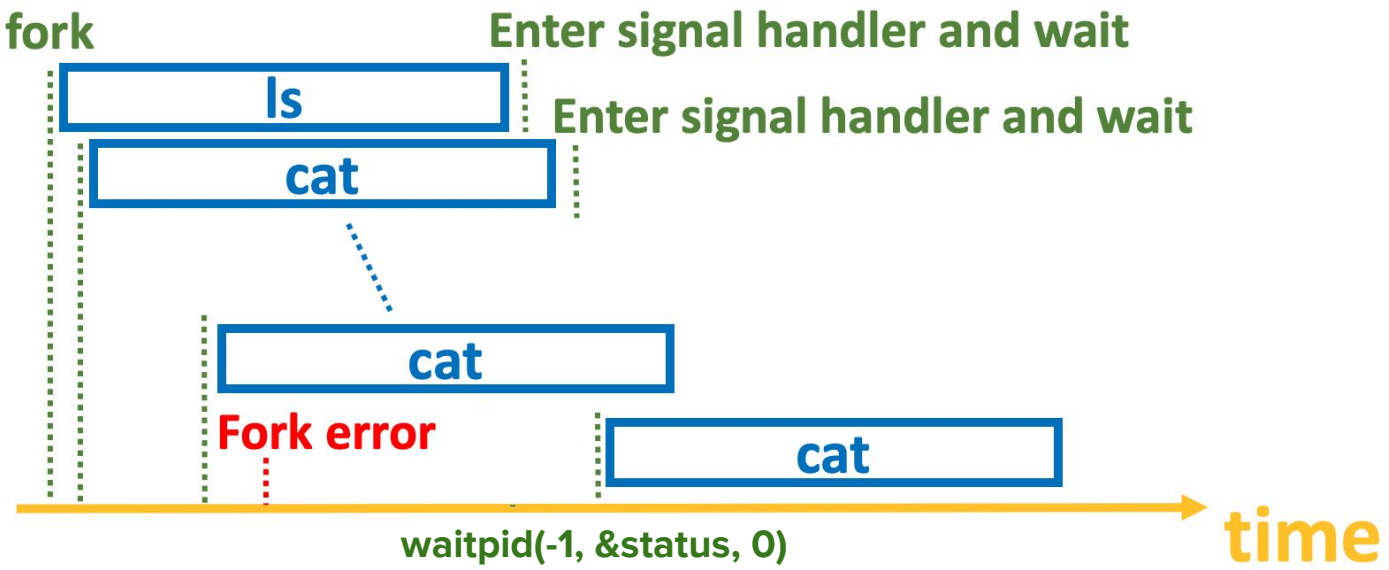
```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
      waitpid(-1, &status, 0);
}

……

If ( ! lineEndsWithPipeN) {
      for (pid in pidTableForCurrentLine) {
            int status;
            waitpid(pid, &status, 0);
      }
}
```

```
Void childHandler(int signo) {
      int status;
      while (waitpid(-1, &status, WNOHANG) > 0) {
            //do nothing
      }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

Implementation 7 : Signal Handler

ls | cat | cat | cat | cat ..... cat | cat



Implementation 7 : Signal Handler

ls | cat | cat | cat | cat ..... cat | cat

```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
      waitpid(-1, &status, 0);
}

……

If ( ! lineEndsWithPipeN) {
      for (pid in pidTableForCurrentLine) {
            int status;
            waitpid(pid, &status, 0);
      }
}
```

```
Void childHandler(int signo) {
      int status;
      while (waitpid(-1, &status, WNOHANG) > 0) {
            //do nothing
      }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

Implementation 7 : Signal Handler

ls | cat | cat | cat | cat ..... cat | cat

```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
        waitpid(-1, &status, 0);
}

……
```

```
If ( ! lineEndsWithPipeN) {
        for (pid in pidTableForCurrentLine) {
                int status;
                waitpid(pid, &status, 0);
        }
}
```

```
void childHandler(int signo) {
        int status;
        while (waitpid(-1, &status, WNOHANG) > 0) {
                //do nothing
        }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

Do we need both of this ?

Implementation 7 : Signal Handler

ls | cat | cat | cat | cat ..... cat | cat

```
signal(SIGCHLD, childHandler);
……
while (fork() < 0) {
      waitpid(-1, &status, 0);
}

……
```

```
If ( ! lineEndsWithPipeN) {
      for (pid in pidTableForCurrentLine) {
            int status;
            waitpid(pid, &status, 0);
      }
}
```

```
void childHandler(int signo) {
      int status;
      while (waitpid(-1, &status, WNOHANG) > 0) {
            //do nothing
      }
}
```

-1: wait for any child process
WNOHANG: return immediately if no child has exited.

This is for solving % ordering, we need to keep this.

Implementation 7: Signal Handler

```
% cat largeFile.txt | number
largeFile.txt content...
% ls
bin    test.html
%
```

```
void childHandler(int signo) {
   int status;
   while (waitpid(-1, &status, WNOHANG) > 0) {
      //do nothing
   }
}
```

```
If ( ! lineEndsWithPipeN) {
   for (pid in pidTableForCurrentLine) {
      int status;
      waitpid(pid, &status, 0);
   }
}
```