CS350 Lab 2: TopSay (Pipes)

Xiaoshuang Wang
Computer Science Dept.
Binghamton University

Outline

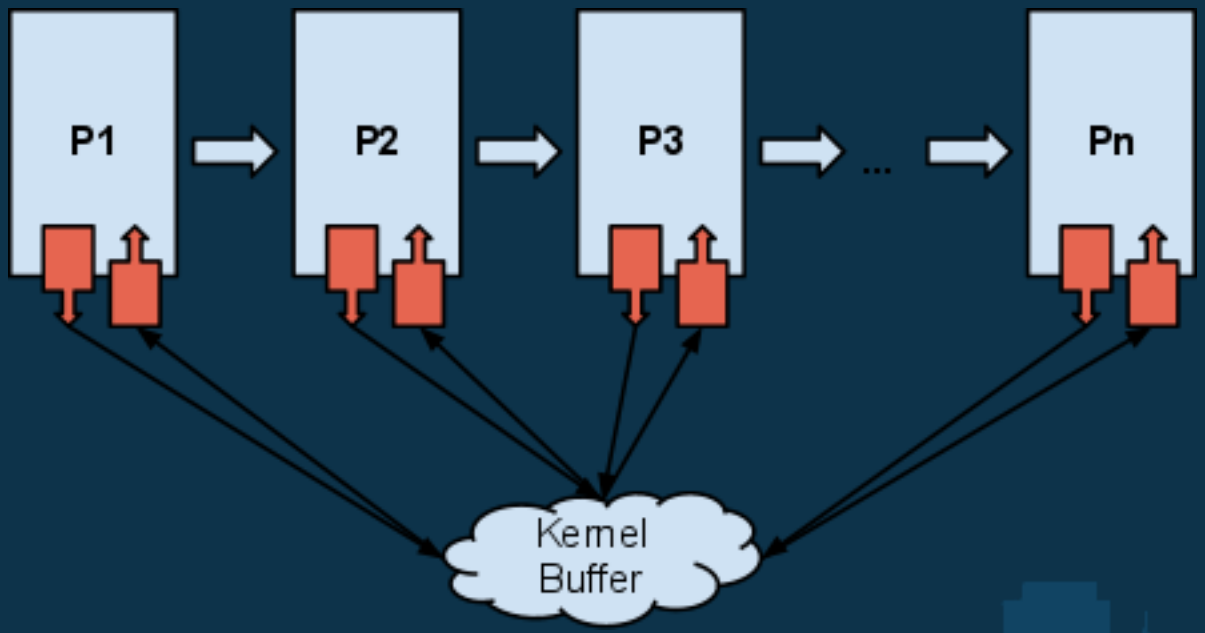
- Problem Statement
- Problem Analysis

Problem Statement

Write a program "topSays", which takes an integer argument N. You may assume N to be less or equal to 5. Your program works as follow:
Top parent - forks ==> child1 - forks ==> child1's child - forks ... ==> Nth generation child

The top parent will then pass a message to the child at the lowest level (Nth generation child) through a pipe, and that child should print the received message to stdout.
Bonus
Support arbitrary value of N (Any integer value greater or equal to 1).
Hint: Use recursive function.

Problem Analysis



Problem Analysis

Parent process:

1. Create pipe (Do this first before fork, Why?)
2. Somehow create all the children.
    1. ... *something here*
3. Close the pipe read descriptor (Why?)
4. Write something to the pipe.
5. Wait the child's termination.

The bottom child:

1. Close the pipe write.
2. Read something from the pipe.
3. Print it out.

What about all the other children???

Example Code

```
#include <sys/wait.h>
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
int
main(int argc, char *argv[])
{
    int pfd[2];
    pid_t cpid;
    char buf;
    assert(argc == 2);
    if (pipe(pfd) == -1) { perror("pipe"); exit(EXIT_FAILURE); }
    cpid = fork();
    if (cpid == -1) { perror("fork"); exit(EXIT_FAILURE); }
    if (cpid == 0) {    /* Child reads from pipe */
        close(pfd[1]);          /* Close unused write end */
        while (read(pfd[0], &buf, 1) > 0)
            write(STDOUT_FILENO, &buf, 1);
        write(STDOUT_FILENO, "\n", 1);
        close(pfd[0]);
        _exit(EXIT_SUCCESS);
    } else {            /* Parent writes argv[1] to pipe */
        close(pfd[0]);          /* Close unused read end */
        write(pfd[1], argv[1], strlen(argv[1]));
        close(pfd[1]);          /* Reader will see EOF */
        wait(NULL);             /* Wait for child */
        exit(EXIT_SUCCESS);
    }
}
```

Thanks