CSED211: Lab 10

Shell Lab2

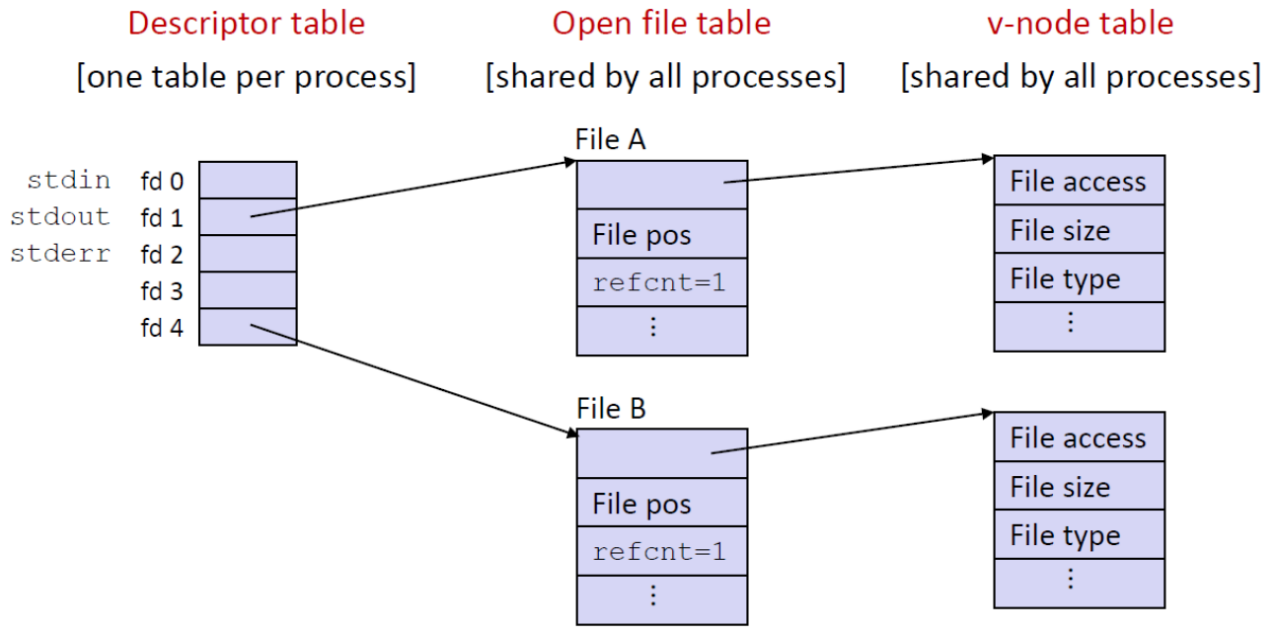
Postech

박재준 : jjpark17@postech.ac.kr

Table of Contents

File Descriptor

- A file descriptor is an abstract indicator used to access a file or I/O resource



Open

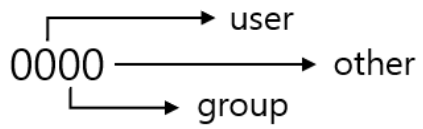
- Open system call
  - open a file, possibly make or delete file
  - open(const char* path, int flags, mode_t mode)

Flags

O_RDONLY	- read only
O_WRONLY	- write only
O_RDWR	- read & write
O_APPEND	- append
O_CREAT	- make file if not exist
O_TRUNC	- delete file if exist

Mode – important when create flag is set

4 - read
2 - write
1 - execute



0760 -> user has all permission, group has read/write permission

open("~~~", O_RDWR | O_CREAT, 0664)

Read & Write

- Read system call
  - read n bytes to the file associated with the file descriptor
  - read(int filedes, const void *buf, size_t nbyte)

- Write system call
  - write n bytes to the file associated with the file descriptor
  - write(int filedes, const void *buf, size_t nbyte)

Dup

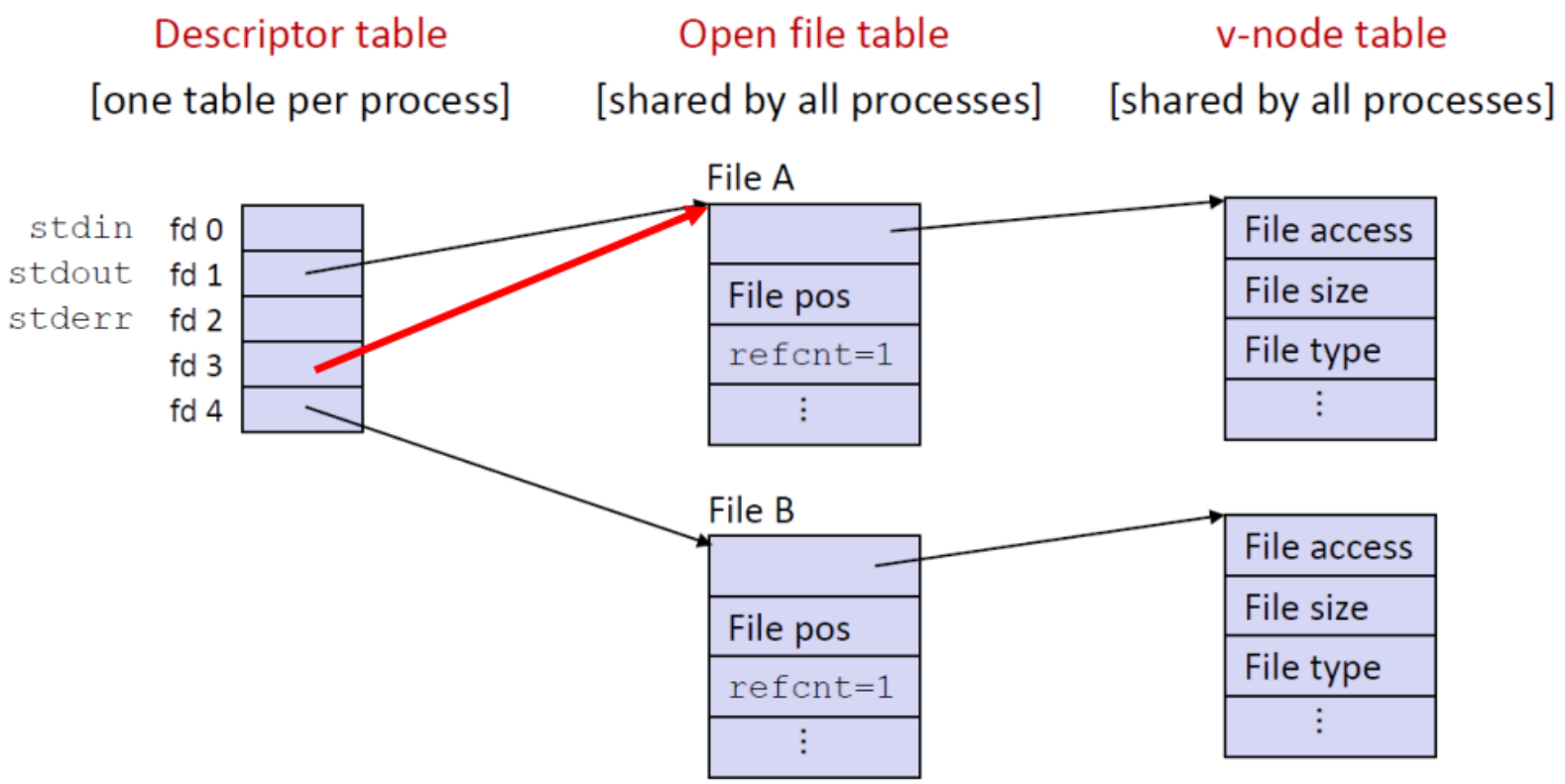
- int dup(int oldfd);
  - Create a copy of the file descriptor oldfd

- int dup2(int oldfd, int newfd);
  - Create a copy of the file descriptor oldfd
  - Assign the copy into the newfd

Example

dup(1);

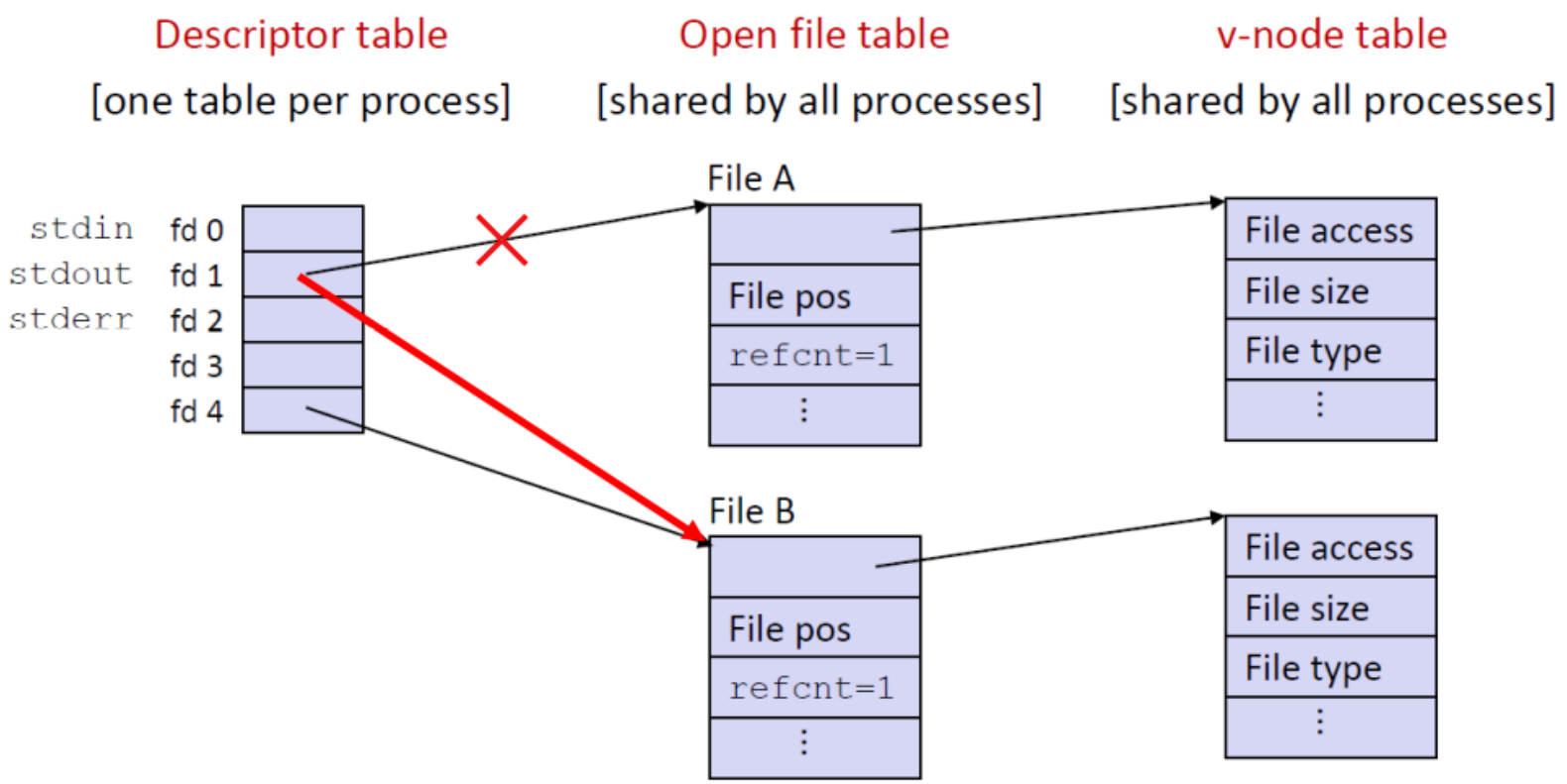
return value : 3



Example

dup2(4, 1);

return value : 1



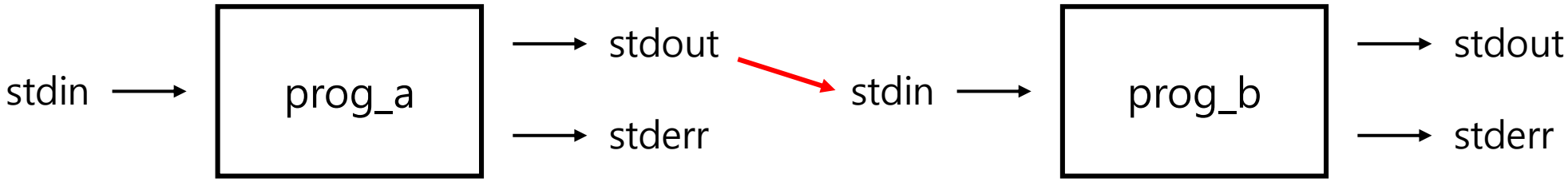
Pipe & Redirection

- The prompt should be the string "`tsh>` ".
- The command line typed by the user should consist of a `name` and zero or more argument rated by one or more spaces. If `name` is a built-in command, then `tsh` should handle it in and wait for the next command line. Otherwise, `tsh` should assume that `name` is the executable file, which it loads and runs in the context of an initial child process (In this c term *job* refers to this initial child process).
- `tsh` need not support pipes (`|`) or I/O redirection (< and >).
- Typing `ctrl-c` (`ctrl-z`) should cause a SIGINT (SIGTSTP) signal to be sent to the cu ground job, as well as any descendents of that job (e.g., any child processes that it forked). no foreground job, then the signal should have no effect.
- If the command line ends with an ampersand `&`, then `tsh` should run the job in the ba Otherwise, it should run the job in the foreground.

Pipe

- Set of process chained together by their standard streams, so that the output text of each process is passed directly as input to the next one
- Command: |

- Ex) ./prog_a | ./prog_b



Redirection

- Set of process redirects streams to user-specified locations
- Command: <, >, >>, …

- prog < file : Read file and use it as stdin
- prog > file : Write stdout to file / if file exist, rewrite
- prog >> file : Write stdout to file / if file exist, append

more info : https://mug896.github.io/bash-shell/redirections.html

Redirection

- Ex) ./prog_a < file_1 > file_2

Practice

```c
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<string.h>

int main()
{
    char buf[256];
    int fd1 = open("1.txt", O_RDWR);
    int fd2 = open("2.txt", O_WRONLY | O_CREAT, 0777);
    int fd3 = dup(fd1);

    scanf("%s", buf);
    write(4, buf, strlen(buf));
    dup2(1, 4);
    write(fd2, buf, strlen(buf));
    return 0;
}
```

1.txt


```
[jjpark17@programming2 practice_quiz]$ gcc practice.c -o practice
[jjpark17@programming2 practice_quiz]$ ./practice
apple
apple[jjpark17@programming2 practice_quiz]$ cat 1.txt
1.txt
[jjpark17@programming2 practice_quiz]$ cat 2.txt
apple[jjpark17@programming2 practice_quiz]$ vim practice.c
[jjpark17@programming2 practice_quiz]$ ./practice > 3.txt
asdf
[jjpark17@programming2 practice_quiz]$ cat 3.txt
asdf[jjpark17@programming2 practice_quiz]$ vim 3.txt
```

Quiz1

- What is the result of the below command when 1.txt doesn't exist?

```
[jjpark17@programming2 practice_quiz]$ ./practice > 3.txt
asdf
```

- A) "asdf" is written in 2.txt only
- B) "asdf" is written in 3.txt only
- C) "asdf" is written in 2.txt and 3.txt
- D) The program stopped with an error

Quiz2

- What is the result of the below command when 1.txt doesn't exist?

```
[jjpark17@programming2 practice_quiz]$ ./practice > 3.txt > 4.txt < 1.txt
```

1.txt

- A) "1.txt" is written in 3.txt
- B) "1.txt" is written in 4.txt
- C) Nothing is written in 3.txt and 4.txt
- D) The program stopped with an error