

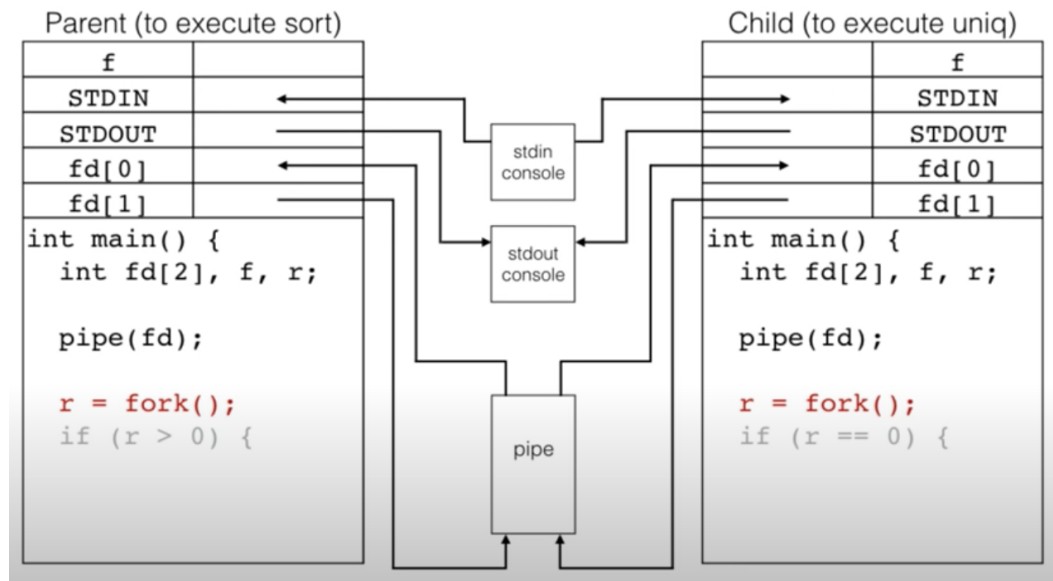
CSC209 Week 11 Notes

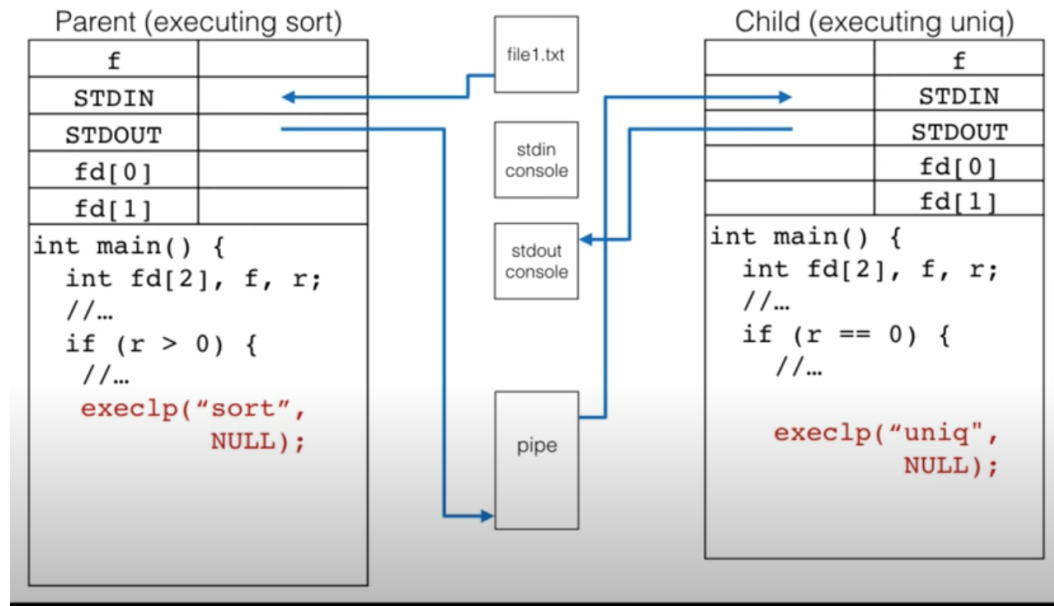
Hyungmo Gu

May 17, 2020

Processes 7 of 8

- Implementing the Shell Pipe Operator
 - piping
 - * purpose is to send output from one end to input of another
 - * done by using *dup2* and *pipe*
 - * *fd[0]* and *fd[1]* must be closed after *dup2*
 - *dup2*
 - * sets up redirection from *fdes* to *fdes2*
 - * **Syntax:** `int dup2(int fdes, int fdes2)`
 - **fdes:** The source file descriptor
 - **fdes2:** The destination file descriptor
 - Example
 - * Executing *sort* in parent and passing it to child for *uniq*
 - This is the same as ‘*sort <file1 | uniq*’





```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <fcntl.h>
6  #include <unistd.h>
7  #include <sys/wait.h>dd
8
9  // equivalent to sort < file1 | uniq
10 void sort_by_parent(int *fd);
11 void uniq_by_child(int *fd);
12
13 int main() {
14     int fd[2], r;
15
16     if ((pipe(fd) == -1)) {
17         perror("pipe");
18         exit(1);
19     }
20
21     r = fork();
22
23     if (r < 0) {
24         perror("fork");
25         exit(1);
26     }
27
28     if (r > 0){
29         sort_by_parent(fd);
30     } else {
31         uniq_by_child(fd);
32     }
33 }
34
```

```
35 void sort_by_parent(int *fd) {
36     int filedes = open("file1.txt", O_RDONLY);
37
38     // reconfigure so all input from file1 are redirected to
39     stdin
40     if (dup2(filedes, fileno(stdin)) == -1) {
41         perror("dup2.1");
42         exit(1);
43     }
44
45     // reconfigure so all output from stdout is redirected to
46     write part of pipe
47     // this is to sent to uniq
48     if (dup2(fd[1], fileno(stdout)) == -1) {
49         perror("dup2.2");
50         exit(1);
51     }
52
53     // close read part of pipe
54     if (close(fd[0]) == -1) {
55         perror("close1");
56     }
57
58     // close write since it's redirected to stdout
59     if (close(fd[1]) == -1) {
60         perror("close2");
61     }
62
63     // close file since it won't be used directly
64     if (close(filedes) == -1) {
65         perror("close3");
66     }
67
68     // executes terminal's sort
69     execl("/usr/bin/sort", "sort", (char *) 0);
70 }
71
72 void uniq_by_child(int *fd) {
73     // reconfigure stdin so that it reads from pipe
74     if (dup2(fd[0], fileno(stdin))) {
75         perror("dup2");
76         exit(1);
77     }
78
79     // close the write pipe (see diagram)
80     if (close(fd[1]) == -1) {
81         perror("close");
82     }
83
84     // close the read pip since it will be read from stdin of
85     pipe
86     if (close(fd[0]) == -1) {
87         perror("close");
88     }
89 }
```

```
86     execl("/usr/bin/uniq", "uniq", (char*) 0); // <- execute
    file from file path, and return 0 if successful
87     fprintf(stderr, "ERROR: exec should not return\n"); // <-
    run if uniq not run
88 }
89
```

Listing 1: pipe_example_1.c

```
1     >>> gcc -Wall pipe_example_1.c
2     >>> ./a.out
3     Fail T1
4     Fail T2
5     Fail T5
6     Pass
7
```

- **&:** Allows the program to run simultaneously

```
1     prog1 & prog2; #<- prog1 and prog2 runs in parallel
2     prog1 & # <- prog1 runs in background, and cursor is returned
    immediately
3
```

- **#!:** Is PID of latest program that's running in the background