Exec, Pipe and Dup

Computer Architecture & OS Lab
Dept. of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Exec

- System calls that allow a process to execute a specified program
  - Process identifier remains the same.
  - There is no return from exec.

Sample program: execlp.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ipc.h>

main()
{
    execlp("cal","cal","2001",NULL);
    printf("This statement is not executed
                                    if execlp succeeds.\n");
}
```

Pipe

- The pipe() system call
  - Creates a pipe that can be shared between processes
  - It returns two file descriptors,
    - One for reading from the pipe
    - The other, for writing into the pipe

Using pipe: pipe.c

```
#include <stdio.h>
#include <unistd.h>     /* Include this file to use pipes */
#define BUFSIZE 80

main()
{
    int fd[2], n=0, i;
    char line[BUFSIZE];

    pipe(fd);  /* fd[0] is for reading,
                          fd[1] is for writing */
```

Using pipe: pipe.c

```
    if (fork() == 0) {
            close(fd[0]);     /* The child will not read */
            for (i=0; i < 10; i++) {
                    sprintf(line,"%d",n);
                    write(fd[1], line, BUFSIZE);
                    printf("Child writes: %d\n",n);  n++; sleep(2);
    }}
    else {
            close(fd[1]); /* The parent will not write */
            for (i=0; i < 10; i++) {
                    read(fd[0], line, BUFSIZE);
                    sscanf(line,"%d",&n);
                    printf("\t\t\t Parent reads: %d\n",n);
}   }}
```

Dup

- The dup(fd) system call:
  - Copies the descriptor, fd, into the first empty slot in the file descriptor table of the process
  - Recall that the 0^{th} location of the FD table is for stdin and the 1^{st} location of the FD table is for stdout.
  - We can use this information to use close() and dup() for redirecting stdin and/or stdout.

Sample program: dup.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ipc.h>

main()
{
    int fd[2], n=0, i;

    pipe(fd);
```

Sample program: dup.c

```
if (fork() == 0) {          /* Child process */

        close(1) ; dup(fd[1]) ; /* Redirect the stdout of this
                                   process to the pipe. */
        close(fd[0]);
        for (i=0; i < 10; i++) { printf("%d\n",n);  n++; }
    }
    else {                  /* Parent process */

        close(0) ; dup(fd[0]) ; /* Redirect the stdin of this
                                   process to the pipe */
        close(fd[1]);
        for (i=0; i < 10; i++) {  scanf("%d",&n);
                printf("n = %d\n",n); sleep(1);  }
} }
```

Assignment

- Write a program that does the following:
  - It prompts the user to enter the coordinates (x,y) of a set of points terminated by –1.
  - It then forks a child process that executes the gs (ghostscript) program.
  - The parent reads in one point at a time and generates a postscript command that draws a line from the previous point.
  - The postscript command is passed to the child process.
  - When the user enters –1, the parent asks the child to quit and then terminates.

Drawing Lines with Postscript

```
250 250 translate
0 0 moveto
100 100 lineto
100 200 lineto
closepath
stroke
quit
```