

CSI 402 – Systems Programming – Handout 14.3

An Example to Illustrate an Inherited File Descriptor

Note: The following example is taken from pages 100–101 of the text by Haviland et al. It shows that a child process inherits file descriptors from its parent. The parent opens a data file and then forks a child. The child also accesses the same file. The program shows that changes made to the file position by the child are visible to the parent and vice versa.

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>

#define BUF_SIZE 10

void print_pos(const char *, int); /* Prototype. */

int main(void) {

    /* This function creates a child process; the parent and the child */
    /* access the same file. */

    int fd; /* File descriptor. */
    pid_t child; /* Child pid returned by fork. */
    pid_t c; /* Pid of child to be returned by wait. */
    char buf[BUF_SIZE]; /* Buffer for reading from file. */
    int j; /* Temporary. */

    /* Open the data file. */
    if ((fd = open("sample.dat", O_RDONLY)) == -1) {
        fprintf(stderr, "Couldn't open file sample.dat\n"); exit(1);
    }

    /* Advance the file pointer by reading from the file and make sure */
    /* that the read was successful. */
    if((j = read(fd, buf, BUF_SIZE)) < BUF_SIZE) {
        fprintf(stderr, "File sample.dat does not have enough data.\n"); exit(1);
    }

    /* Print position in file before child is created through fork. */
    print_pos("Before fork", fd);
```

(over)

```

/* Fork child and wait for child to complete. */
switch ((child = fork())) {

    case -1: /* Fork failed. */
        fprintf(stderr, "Fork failed.\n"); exit(1);

    case 0: /* Child process. */
        if((j = read(fd, buf, BUF_SIZE)) < BUF_SIZE) {
            fprintf(stderr, "Child: sample.dat does not have enough data.\n");
            exit(1);
        }
        print_pos("Child after read", fd); break;

    default: /* Parent process. */
        c = wait(NULL); /* Wait for child to complete. */
        print_pos("Parent after wait", fd); break;

} /* End of switch. */
return 0;
} /* End of main. */

void print_pos(const char *s, int fd) {
    /* Print the position in the file. */
    off_t pos;

    if ((pos = lseek(fd, 0, SEEK_CUR)) == -1) {
        fprintf(stderr, "Call to lseek failed.\n"); exit(1);
    }
    printf("%s: %ld\n", s, pos);
} /* End of print_pos. */

```

Output:

```

Before fork: 10
Child after read: 20
Parent after wait: 20

```