Embedded Operating Systems

Agenda

- IO Redirection
- Pipe

IO Redirection

- Shell provides redirection feature, which which input, output and/or error of the process can be redirected into the file instead of terminal.
  - terminal> command < in.txt
  - terminal> command > out.txt
  - terminal> command 2> err.txt
- redirection can be done programmatically using dup() system call.

dup() syscall

- Copies given file descriptor on lowest numbered available file descriptor.
- dup(fd)
  - arg1: Copies file descriptor on lowest numbered available file descriptor.

dup2() syscall

- Copies given old file descriptor on given new file descriptor.
- dup2(old_fd, new_fd)
  - arg1: Old file descriptor to be copied.
  - arg2: New file descriptor on which old fd is to be copied. If new_fd is open, it will be first closed and old_fd will be copied on it.

Pipe

- Pipe is used to communicate between two processes.

- It is stream based uni-directional communication.
- Pipe is internally implemented as a kernel buffer, in which data can be written/read.
- If pipe (buffer) is empty, reading process will be blocked.
- If pipe (buffer) is full, writing process will be blocked.
- If writer process is terminated, reader process will read the data from pipe buffer and then will get EOF.
- If reading process is terminated, writing process will receive SIGPIPE signal.
- There are two types of pipe:
    - Unnamed Pipe
    - Named Pipe

Unnamed Pipe

- Used to communicate between related (parent-child) processes.
- terminal> who | wc
    - Internally pipe is created using pipe() syscall.

pipe() syscall

- To create unnamed pipe.
- ret = pipe(fd); // int fd[2];
    - arg1: array of two ints to collect fd (out param).
    - returns 0 on success.
- fd[] gets two ends of pipe.
    - fd[0] -- read end (file descriptor) of pipe
    - fd[1] -- write end (file descriptor) of the file
- Refer diagram for complete flow.

Named Pipe

- Is also called as FIFO
- Used to communicate between unrelated processes.
- Named pipe is created using mkfifo command, which internally calls mkfifo() syscall.

- Fifo is special file. In its inode, type=p and no data blocks are created.

mkfifo() syscall

- To create named pipe.
- ret = mkfifo("fifo path", mode);
  - arg1: path of fifo to be created
  - arg2: fifo mode (permissions)
  - returns 0 on success.
- Refer diagram for complete flow.

Pipe special conditions

- When pipe buffer is full, writer process is blocked.
- When pipe buffer is empty, reader process is blocked.
- If writer fd is closed, reader gets EOF (after reading data present in buffer).
- If reader fd is closed, writer process gets SIGPIPE signal (default action is Terminate).

Assignments

1. Execute "wc" command from your program (using fork() + exec()). The command should read data from the file "in.txt" (instead of terminal) and output should be displayed on terminal.
2. The child process send two numbers to the parent process via pipe. The parent process calculate the sum and return via another pipe. The child process print the result and exit. The parent process wait for completion of the child and then exit.
3. Find the size of pipe buffer in your system.
4. The client process send two numbers to the server process via one fifo. The server process calculate the sum and return via another fifo. The client process print the result.