CSCI 4061: Recitation 6

Today: Review & Pipes

- Review :
  - Fork, wait & exec
  - Dup
  - Directories, Links
  - Stat
- Pipes

fork()

- System call to create a new process (clone of parent)
- Parent and child runs in separate memory spaces
  - Syntax
    - pid_t fork(void);
  - Return values
    - pid > 0 ➔ success, **Parent** execution
    - pid = 0 ➔ success, **Child** execution
    - pid = −1 ➔ failure, no child created

wait()

- Wait for state change in a child of the calling process
- Zombie / orphan process

- Syntax
  - pid_t wait(int *wstatus);
  - pid_t waitpid(pid_t pid, int *wstatus, int options);
- Return values

  Returns process ID of terminated child on success, else returns -1 and errors

exec()

- Deletes the current program state and begins executing a new program.
- If successful, will not return to old program
- Has several variants: execl, execv, execp, execle

Exec Contains Character ...	Meaning
l	Consumes list of args of constant size. (ends with char* NULL)
v	Consumes array of args of variable size.
p	Consumes filename instead of path. Uses default OS 'path' to find file.
e	Overrides default environment (another way to pass args).

dup2()

Makes oldFd now point to newFd in the file descriptor table.

Usecase: stdout, stdin, etc. are default entries in any process's file descriptor table, we can use dup2() to redirect stdin and stdout.

```
#include <unistd.h>

int dup2(int newFD, int oldFD);
```

Stat()

```
#include <sys/stat.h>

int stat(const char *path, struct stat *buf);

int fstat(int fd, struct stat*statbuf);
```

Gets information about a file; `stat()` stats the file pointed to by `path` and fills in `buf` with the information about the file.

Look at man page for stat structure

Directories

- Create/remove
- open/close
- readdir
- chdir
- Getcwd

Links

- Hard link : Alias to the file, essentially an inode reference.
- Symbolic link: Points to another file.

```
#include <unistd.h>

int link(const char *src, const char *link);
int symlink(const char *src, const char *symlink);
```

Pipe()

- Creates a unidirectional data channel.
- pipeFd[0] – Read end, pipeFd[1] – Write end
- Examples

```
#include <unistd.h>

int pipe(int pipeFd[2]);
```

Pipes

- Pipes can be combined with < and >
- Previous example:
  - `echo Be very very quiet. I\'m hunting rabbits. > temp.txt`
  - `sed s/r/w/g < temp.txt > fudd.txt`
- The same can be accomplished with:
  - `echo Be very very quiet. I\'m hunting rabbits. | sed s/r/w/g > fudd.txt`

Pipes – <Non blocking I/O>

- Non-blocking I/O is possible by setting O_NONBLOCK flag.
- Use fcntl() system call to set any flag for Input/Output file descriptor.
  - fcntl(fd[0], F_SETFL, O_NONBLOCK)

Questions?

Exercise

Complete copy.c file.

- It takes 2 cmd line arguments. Input file and output file.
- Main program creates pipe() and calls fork().
- Parent is responsible for reading contents of input file and writing it to pipe.
- Child is responsible for reading from pipe and writing to output file.

Expected Output

```
$ ./copy input.txt output
Parent: 54 chars are read from the input file
Child: 54 chars are written to the output file
```