lab tricks

Taking input :

string
```
char* args = malloc(50);
scanf("%s", args);
execv(args, NULL); ← execv.
```

int
```
int n;
scanf("%d", &n);
```

(int) getpid() ← pid
(int) getppid() ← pid of parent

fork() → returns pid of child for parent, 0 otherwise

wait(NULL); → waits for child to die.

using execv :
```
scanf("%s", args);
rc = fork();
if (rc == 0) execv(args, NULL);
wait(NULL);
```

Using pipe :
```
int fd[2];
int buf;
pipe(fd);
(setup)
```

(sender)
```
int c;
write(fd[1], &c,
      sizeof(c));
```

(Recv)
```
read(fd[0], &buf,
     sizeof(buf));
```

Program states : R (running), D (uninterruptable sleep), S (interruptable sleep), T (stopped), Z (zombie)

Headers : stdio.h, unistd.h, stdlib.h, time.h, sys/types.h, sys/wait.h, math.h, string.h
#include< >

File handling

Open : rd_fd = open("in.txt", O_RDONLY)

```
if (rd_fd == -1){
    //fail.
}
```

write : wr_fd = open("out.txt", O_CREAT|O_RDWR, S_IRWXU); //creates file if not existent

char buff[1];

Transferring data : while (n = read(rd_fd, buff, 1)){
    write(wr_fd, buff, n); }

- C lib f^n like scanf, printf work with modified fd as well.

fd = 0 (STDIN), 1 (STDOUT)

dup2(fd1, fd2) ↦ duplicates fd 1 to fd 2, i.e. fd2 loses original f^n.

status = execvp(comm[0], comm) ↦ comm = list of executable + args

Signal handling : (Quit) → if (SIGNAL == ( )) exit(0);

signal(SIGINT/SIGKILL/SIGTERM, signal_handler) → void signal-handler(int SIGNAL){
    // do whatever
}

SIGINT: ctrl + C

kill(getpid(), SIGTERM);
getpid(): pid
SIGTERM: signal through which it's carried.

chdir(path) → call in the parent process instead of usual execv in child.

Lab Notes