Loading a new program: the execs

```
int  execl (const char *path, const char *argv0, ...,
                                                NULL);

int  execlp(const char * path,
                const char *argv0, ..., NULL);

int  execle (const char *path,
                const char *argv0, ..., NULL,
                const char *env[]);

int  execv (const char *path,
               const char *argv[]);

int  execvp(const char *path,
               const char *argv[]);

int  execve(const char *path,
               const char *argv[],
               const char *env[]);
```

Environment

↳ strings of the form NAME = value

```
char  *getenv(const char *var);
int    setenv (const   char *name,
                const   char *value,
                int     overwrite)

int unsetenv(const char *name)
```

```
$ echo $PATH
   bin;/usr/bin;/usr/local/bin
```

```
execlp("ls", "ls", NULL);
```

```
$ mytalk
```

ls | more

Pipe: two file descriptors connected by a buffer

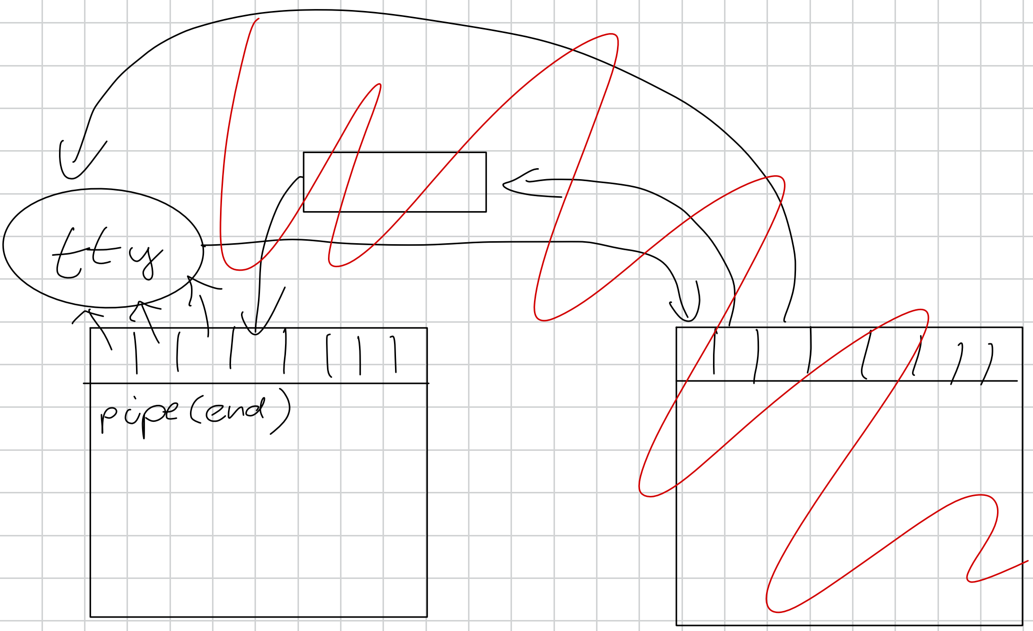
int pipe(int fd[2]);

on success returns ∅

fd[0] open for reading

fd[1] open for writing

on error returns -1



```
# define SIZE 4096
int main(int argc, char *argv[]){
   int num, status, end[2];
   char buffer[SIZE];

   pipe(end);
   if(!(child=fork())){
        /* child */
        dup2(end[1], STDOUT_FILENO);
        close(end[0]);
        close(end[1]);
        execl("/bin/ls" "ls", NULL);
        perror("/bin/ls");
        exit(EXIT_FAILURE);
}else{
  /* parent */
  close(end[1]);
  while((num = read(end[0], buffer, SIZE)) >0)
      write(STDOUT_FILENO, buffer, num);
  close(end[0])
  wait(&status);
  if(WIFEXITED(status))
      exit(WEXITSTATUS(status));
else
   exit(EXIT_FAILURE);
```

```
    return 0;
}
```