I/O redirection :

- redirects[0] = input file (**<**)
- redirects[1] = output file (**>**)
- redirects[2] = append file (**>>**)

- dup() and dup2() :

dup(file_desc_numb) = duplicates file_dec_numb to the lowest available file descriptor number

dup2(file_desc_numb,) = duplicates file_desc_numb to exactly the

dup2() needs a file descriptor number -> open() : opens a file and gives a descriptor number

- open() :

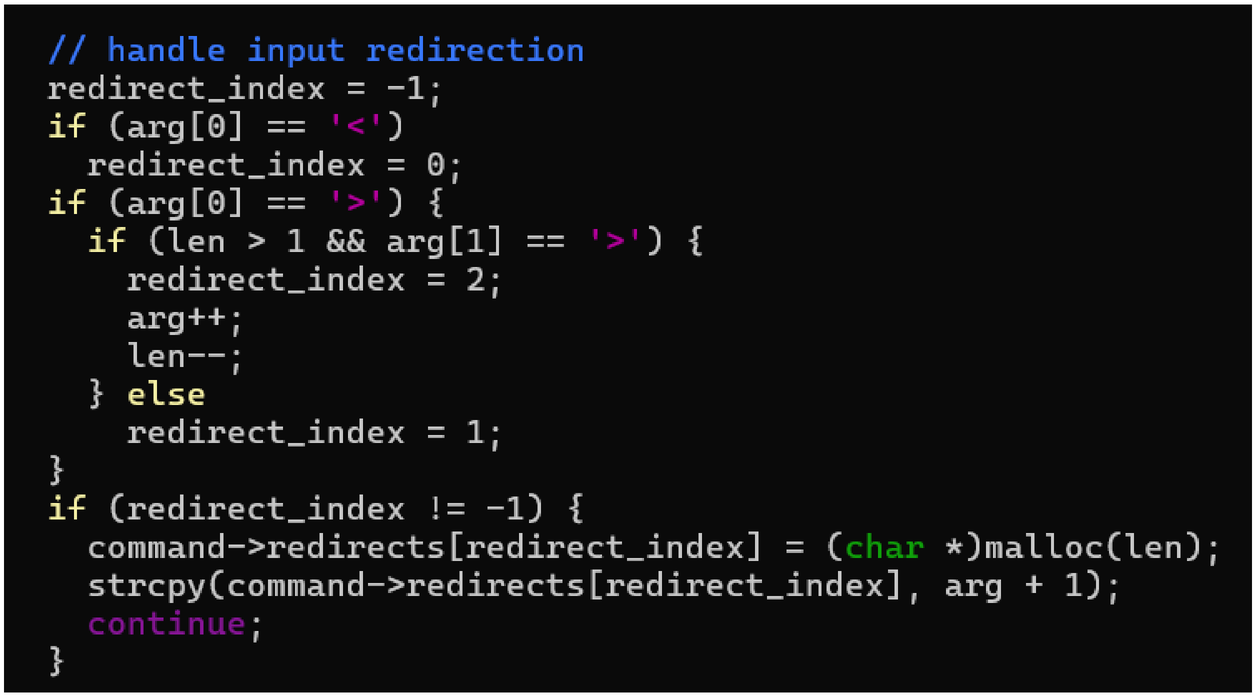
int fd = open("outputfile", flags, permissions);

fd = 2 means file #2 open

Linux has 3 always open file descriptors:

- stdin (keyboard input) : **0**
- stdout (screen output) : **1**
- stderr (error output) : **2**

dup2(fd,1); means make stdout point to fd instead of the screen so anything the program prints now goes to the file specified by fd number.



```
// handle input redirection
redirect_index = -1;
if (arg[0] == '<')
  redirect_index = 0;
if (arg[0] == '>') {
  if (len > 1 && arg[1] == '>') {
    redirect_index = 2;
    arg++;
    len--;
  } else
    redirect_index = 1;
}
if (redirect_index != -1) {
  command->redirects[redirect_index] = (char *)malloc(len);
  strcpy(command->redirects[redirect_index], arg + 1);
  continue;
}
```

parse_command() checks the arg token element's first character to match it to either :

- <
- >
- >>

As I inspected in my **skeleton code analysis ,** command_t struct has a redirects[3] field

redirect_index is set to 0, 1, 2 depending on which symbol matched
strcpy(command->redirects[redirect_index], arg + 1) copies filename into the redirects[index] slot

o_flags for open() functions:

int open(const char *path, int oflag* *)