Project 1 Hints

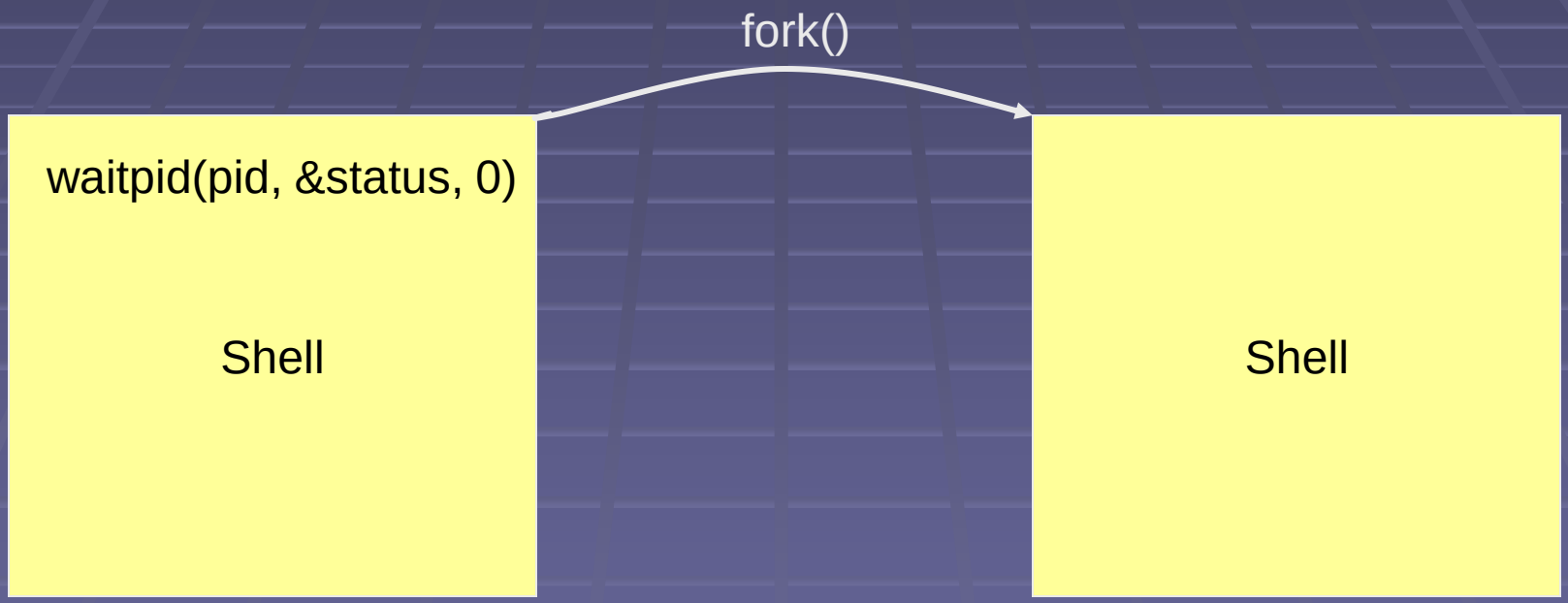
Operating Systems (Lab)

COP 4610 / CGS 5765

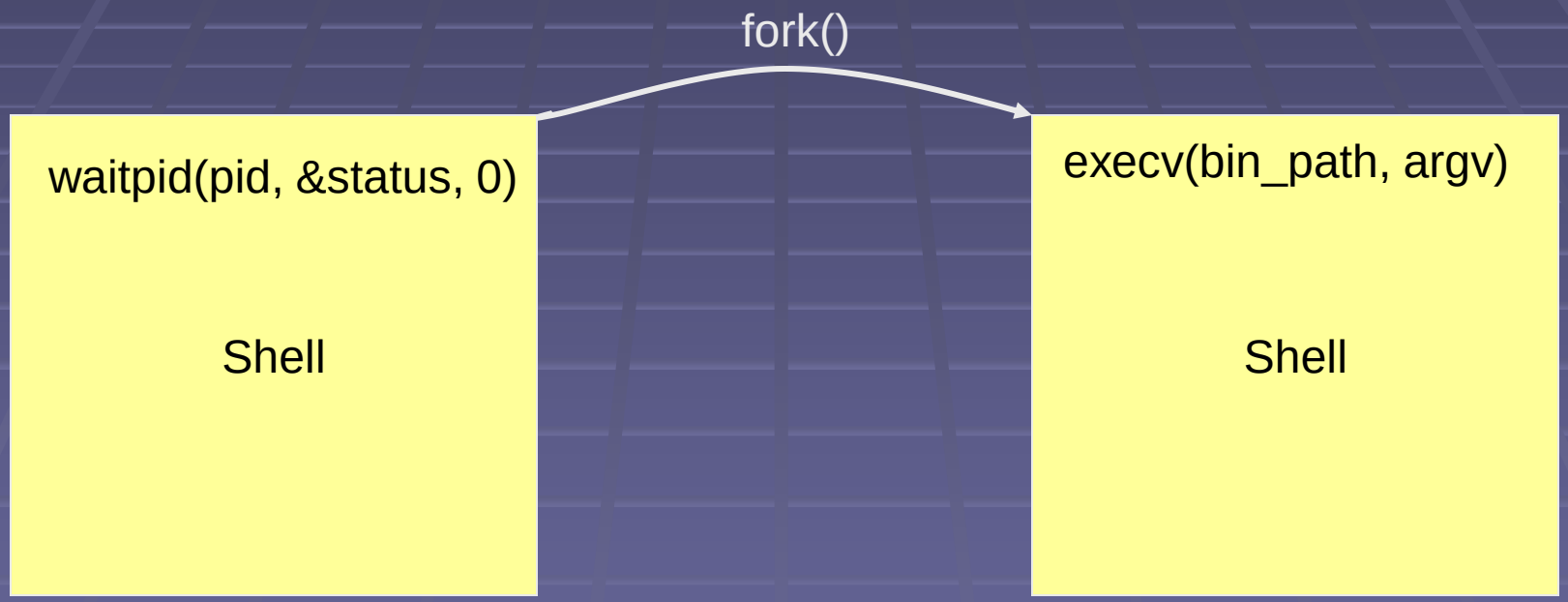
Overall

- Memory allocation
  - calloc()
    - Zero initializes allocation
  - malloc()
    - No zeroing
- User Input
  - fgets()
    - get a string from a stream
- Parsing
    - Use multiple passes to simply parsing
    - One pass to create an array of string tokens
    - One pass for input redirection
    - One pass for output redirection
- chdir() is a system call

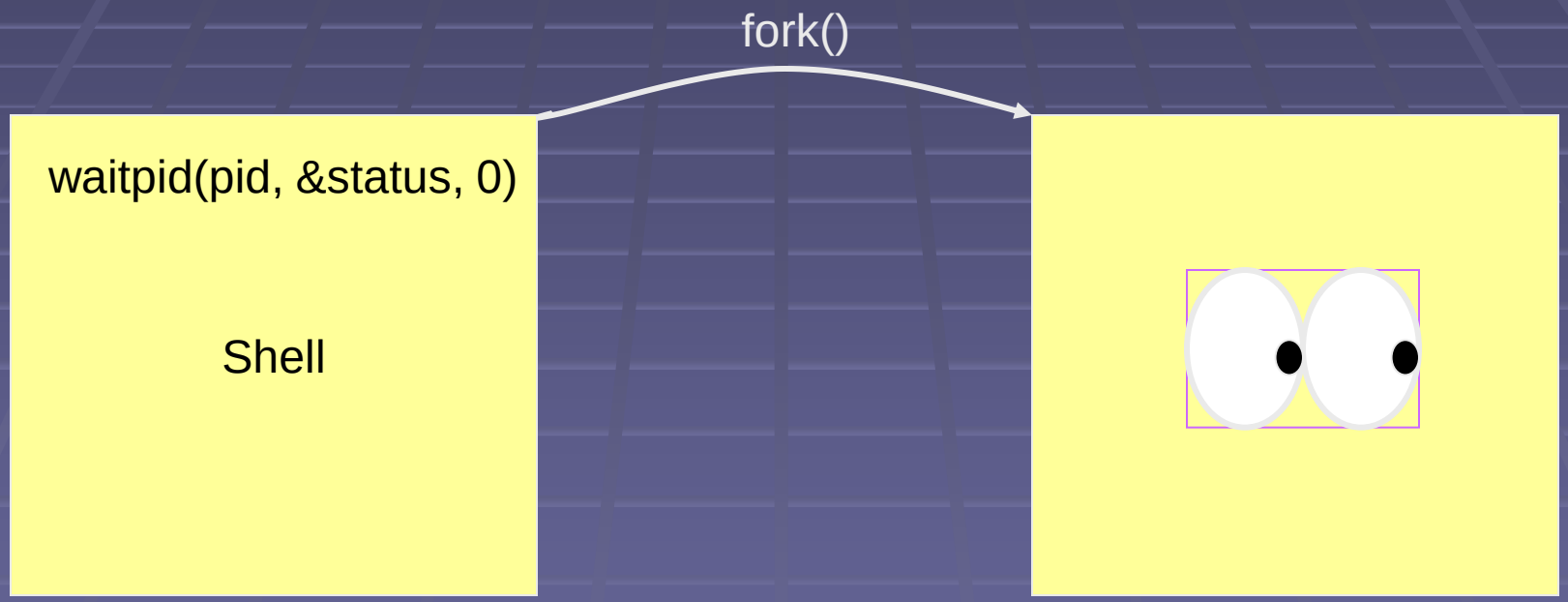
Foreground Execution



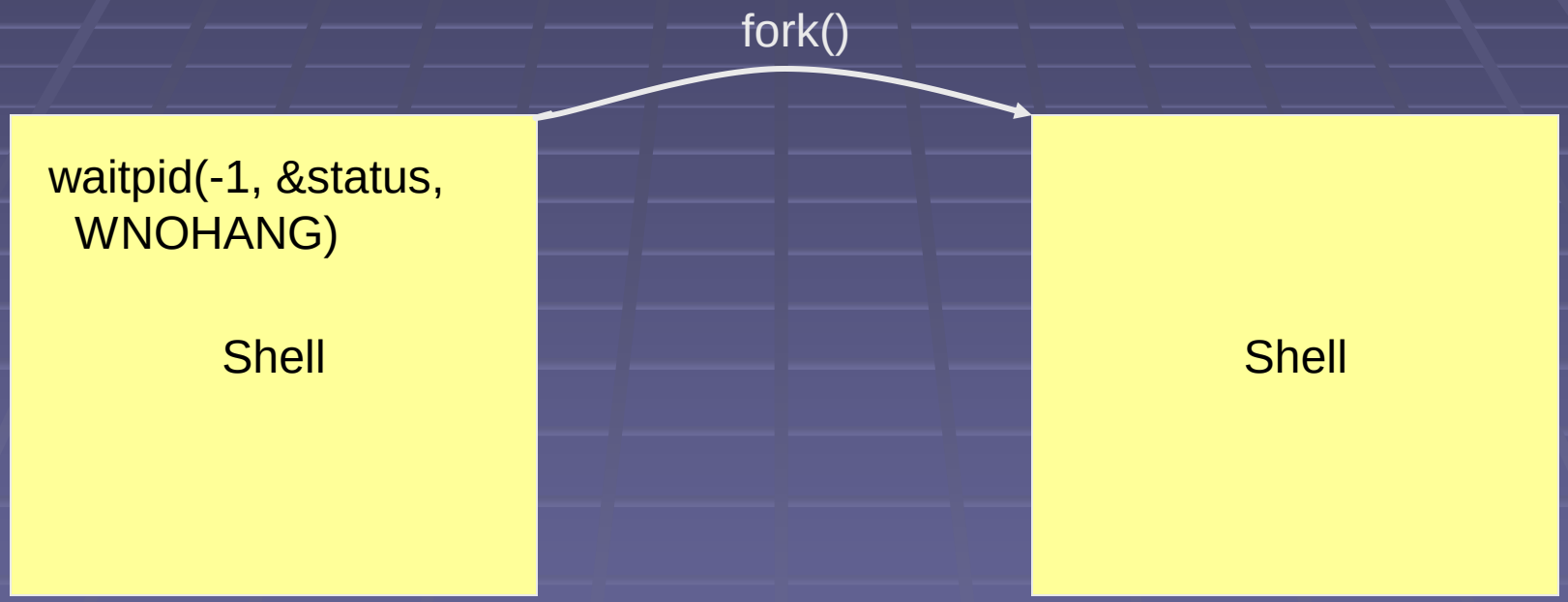
Foreground Execution



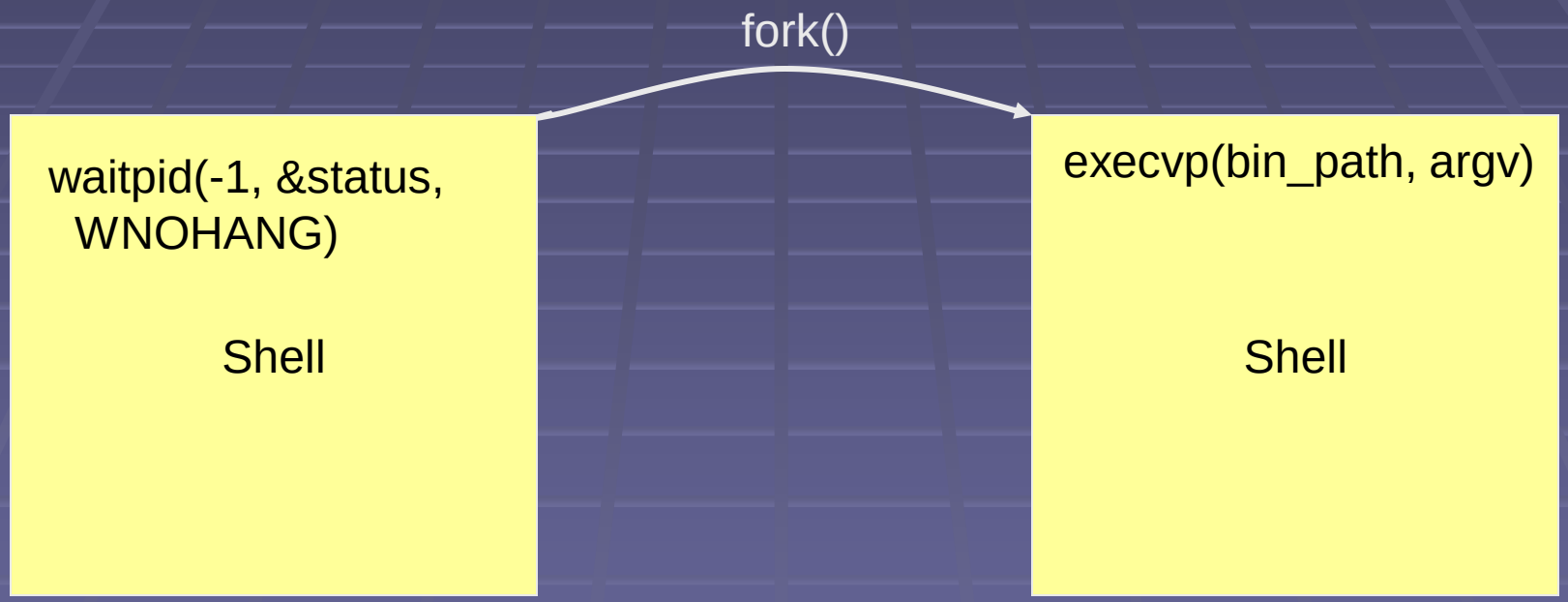
Foreground Execution



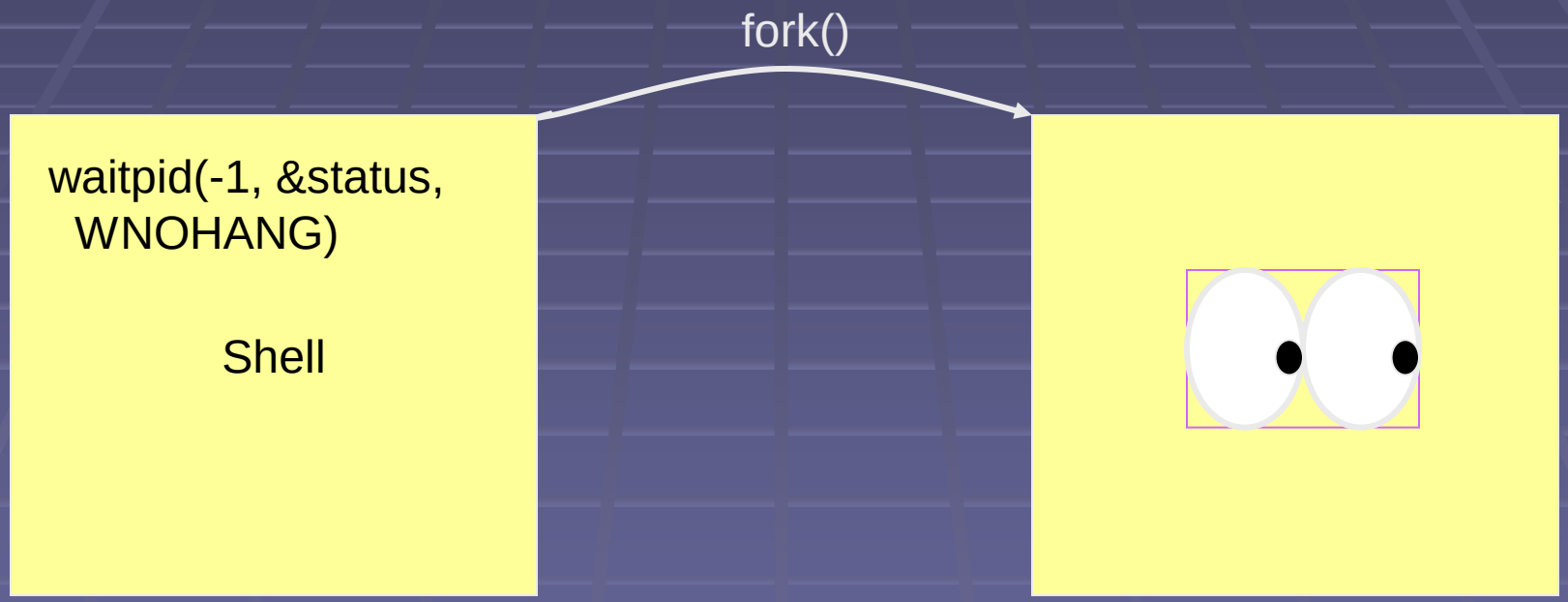
Background Execution



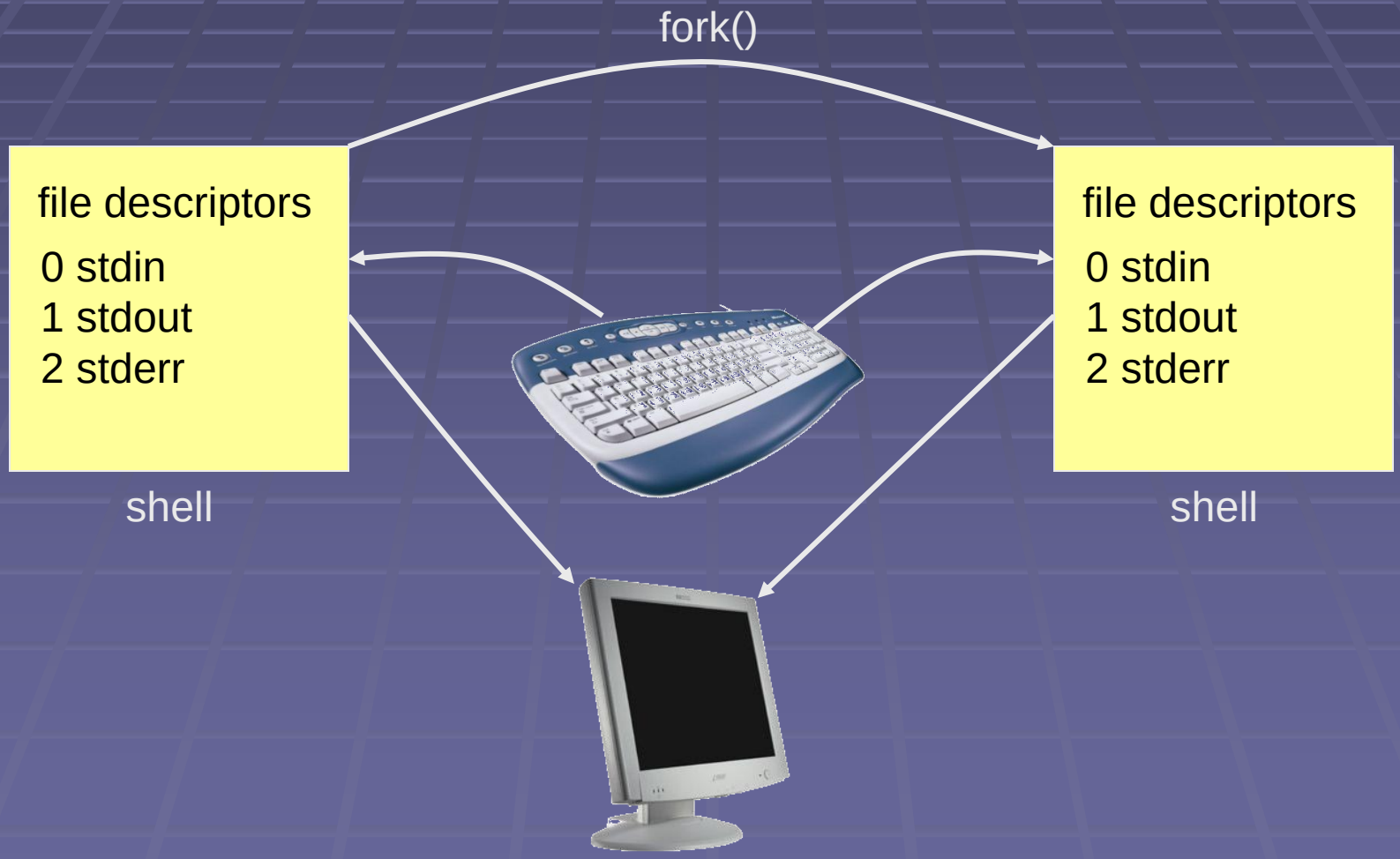
Background Execution



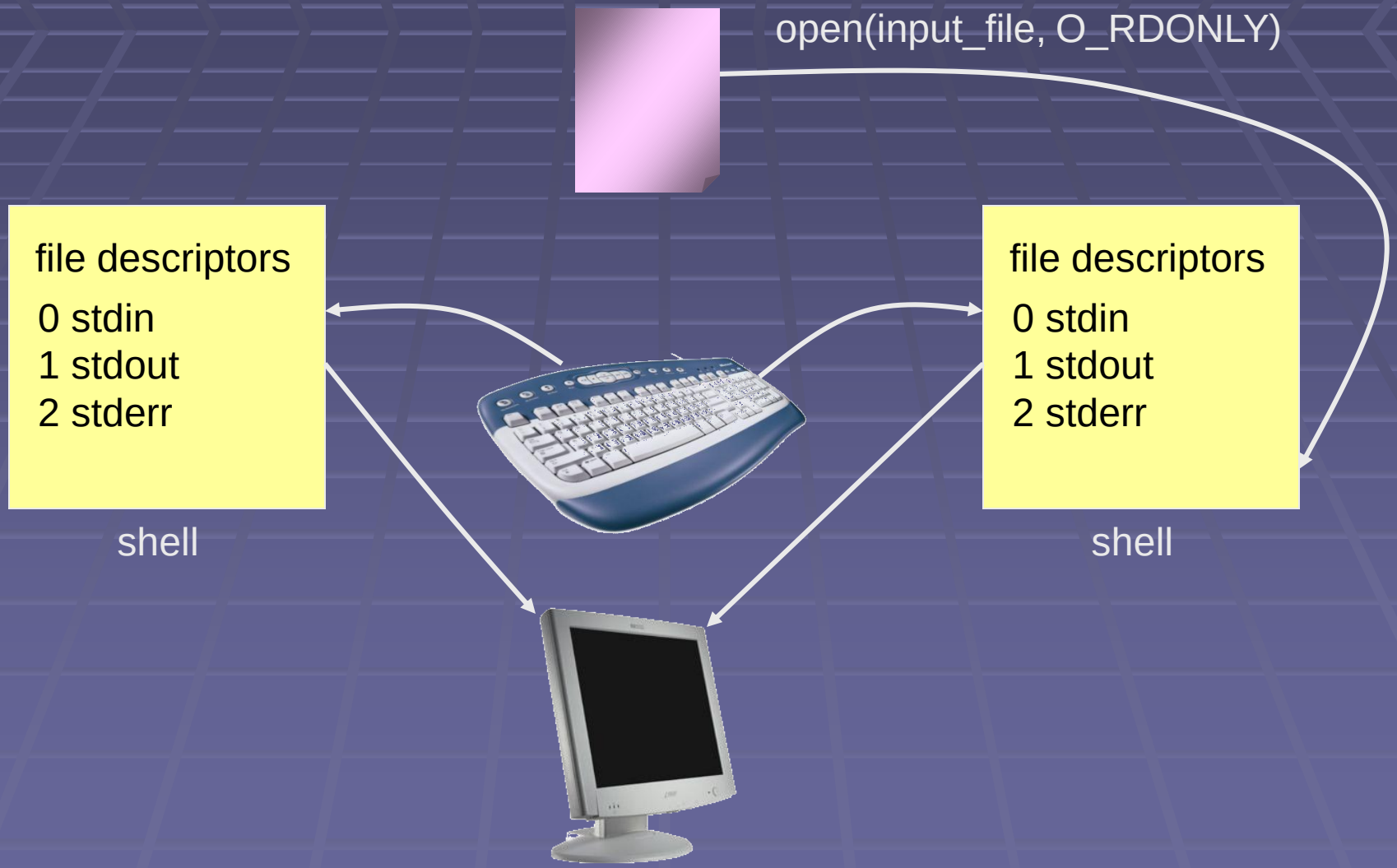
Background Execution



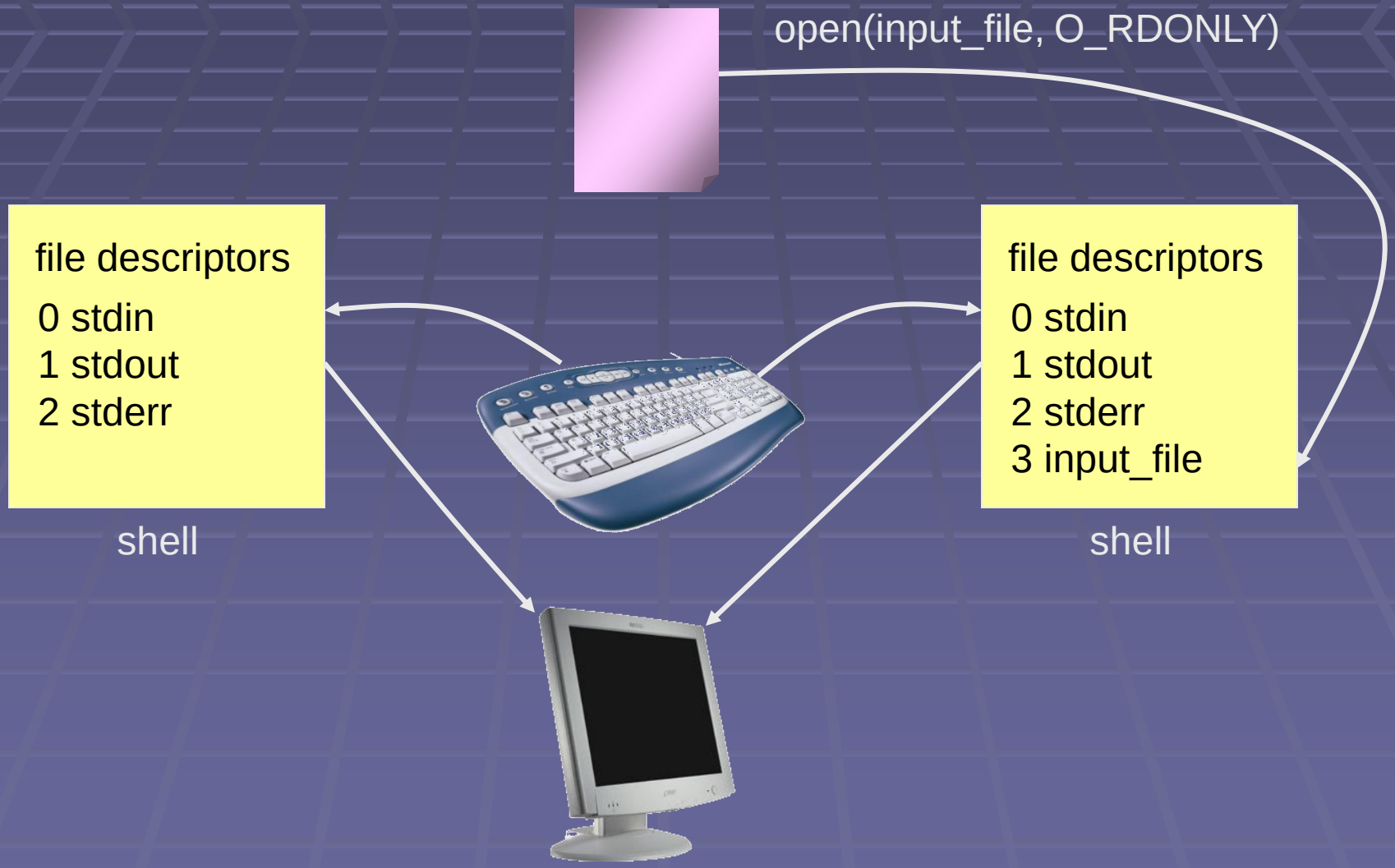
Input Redirection



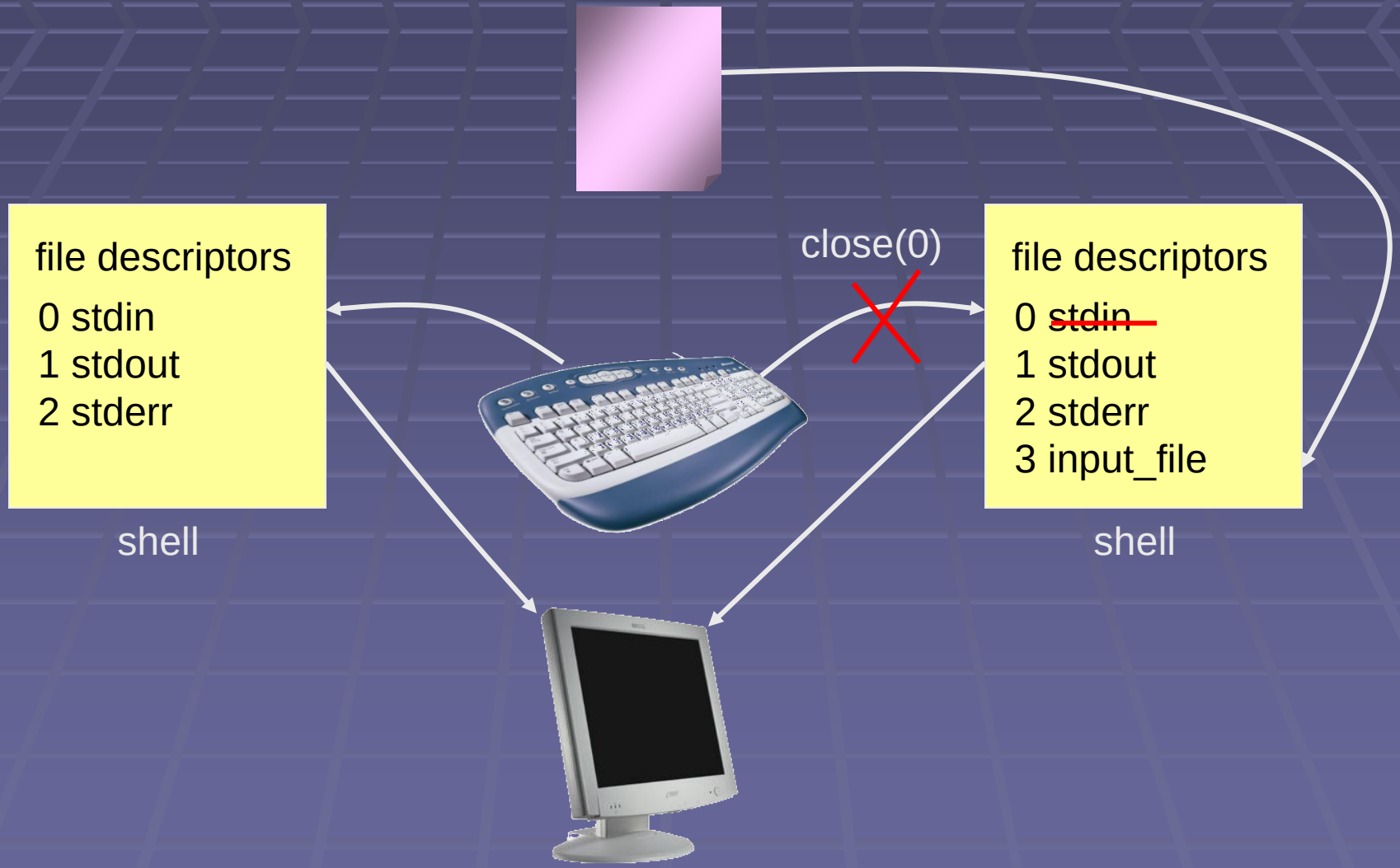
Input Redirection



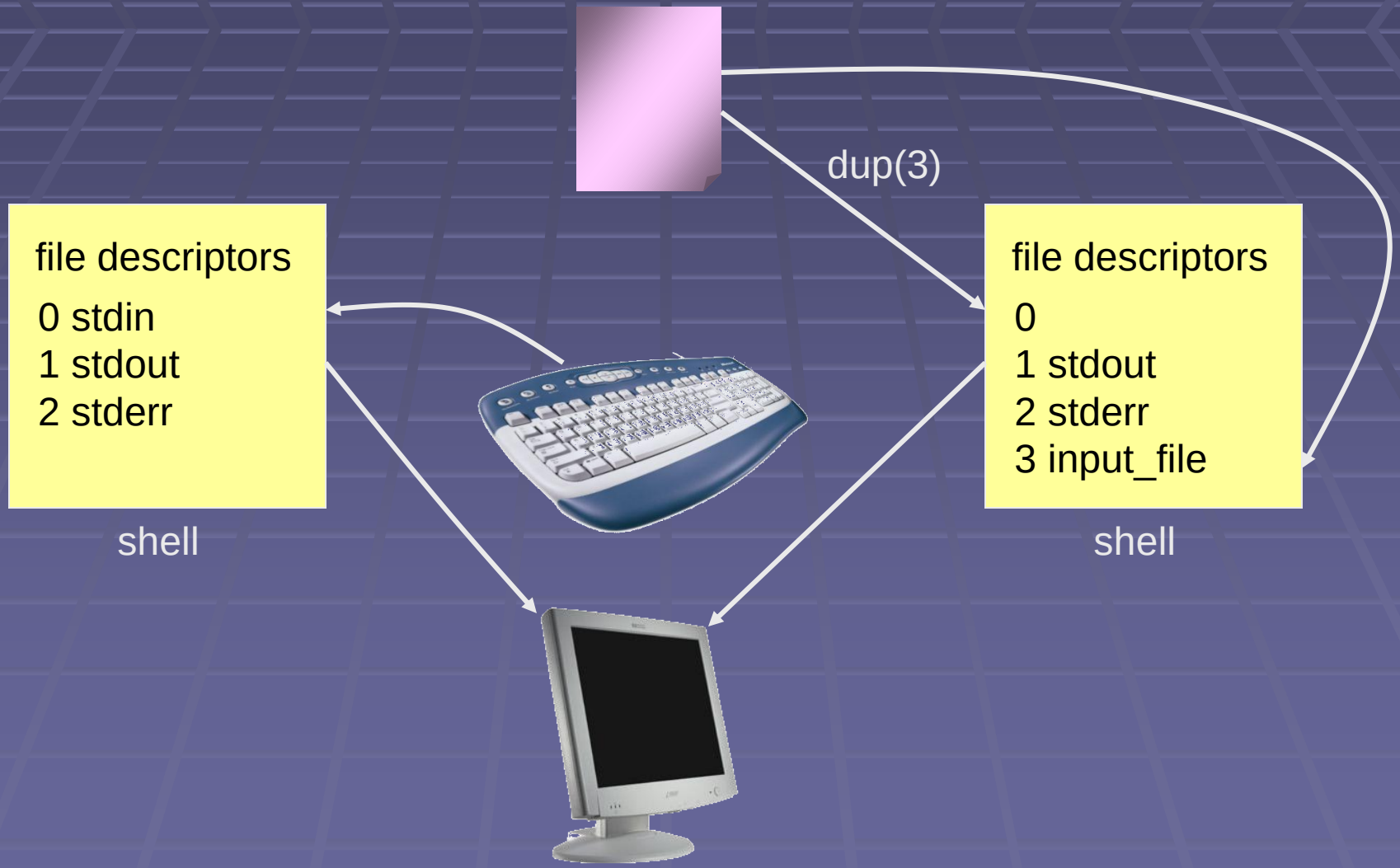
Input Redirection



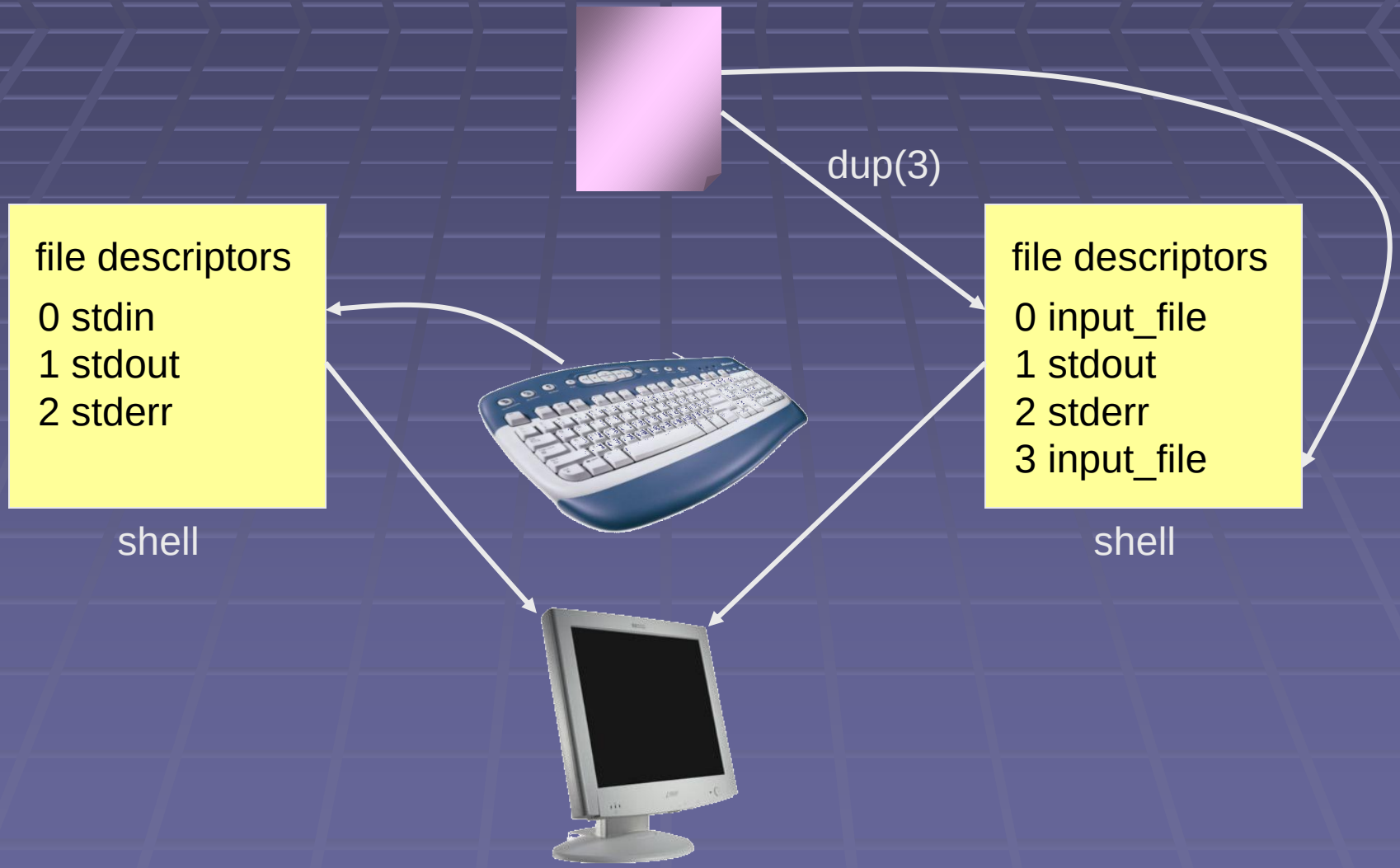
Input Redirection



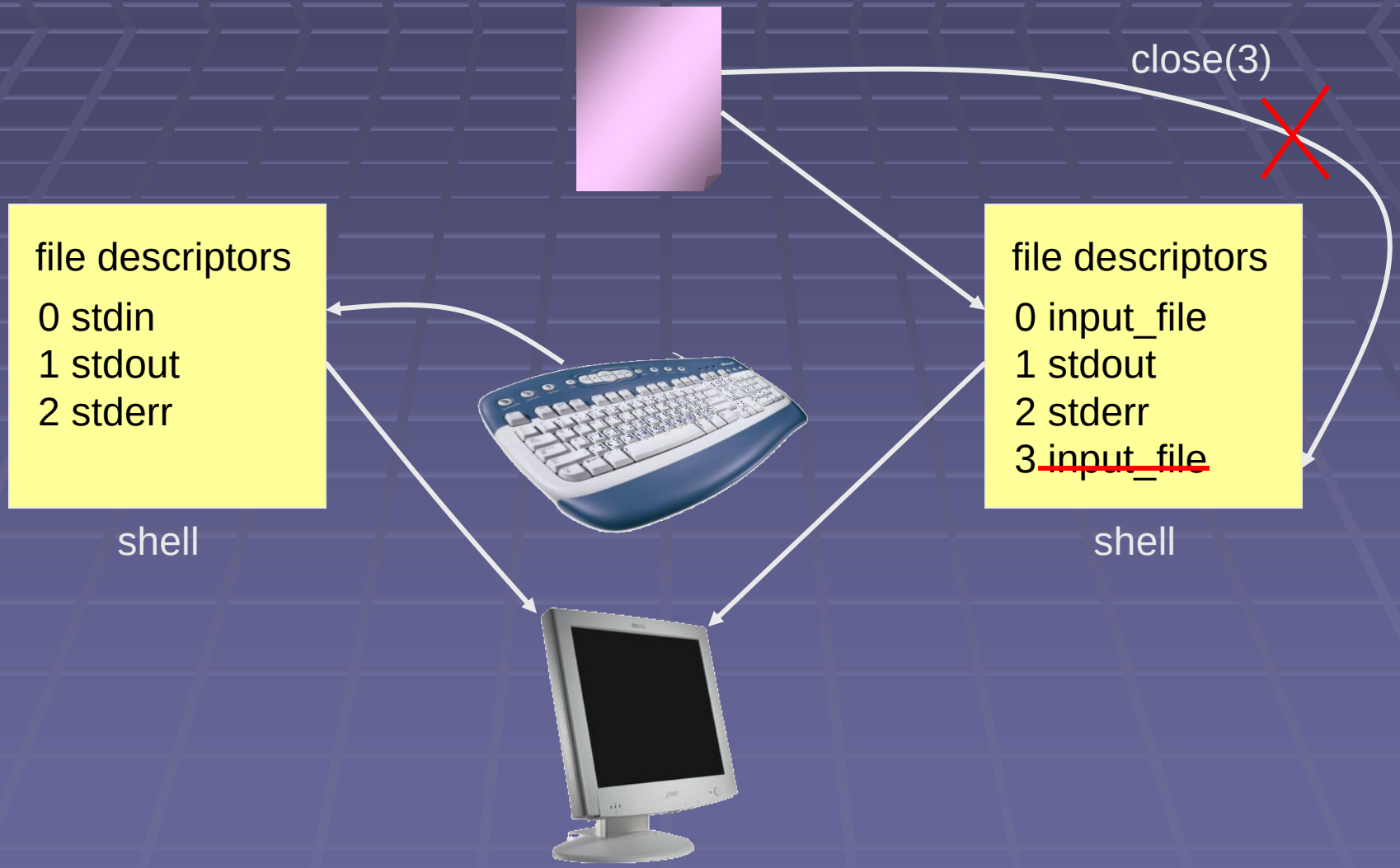
Input Redirection



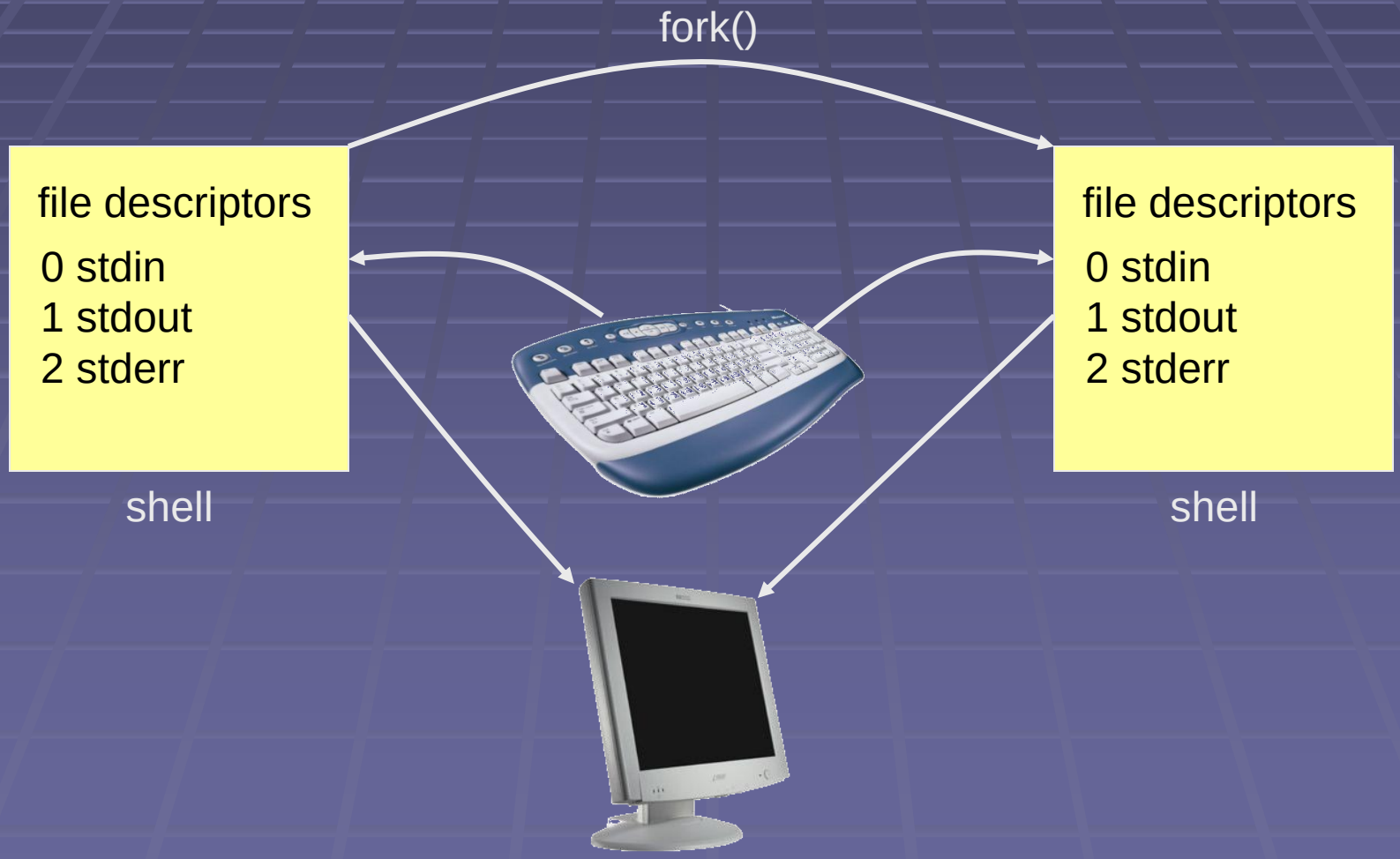
Input Redirection



Input Redirection



Output Redirection



Output Redirection

open(output_file,
O_RDWR | O_CREAT | O_TRUC)

file descriptors
0 stdin
1 stdout
2 stderr

shell

file descriptors
0 stdin
1 stdout
2 stderr

shell

Output Redirection

open(output_file,
O_RDWR | O_CREAT | O_TRUC)

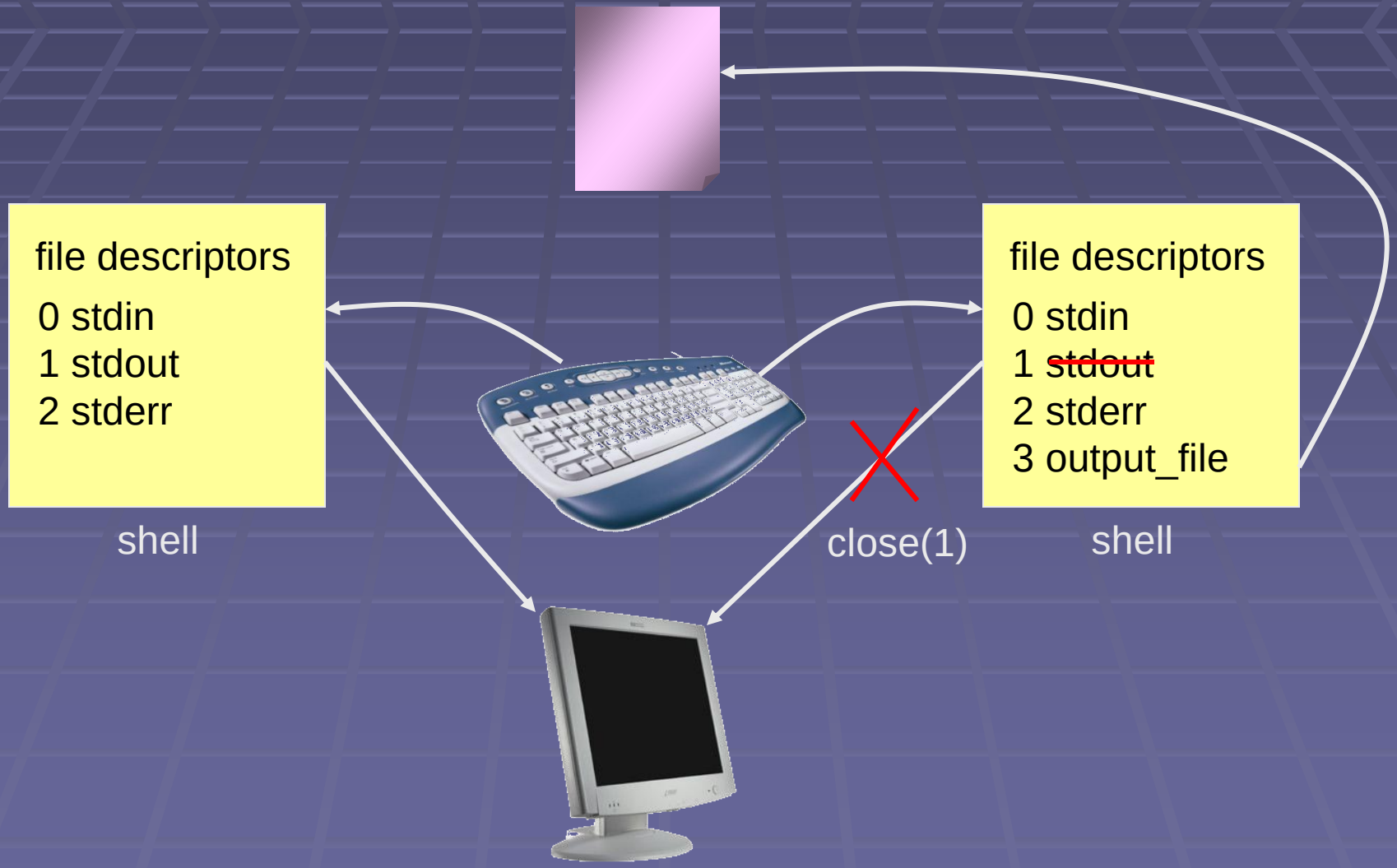
file descriptors
0 stdin
1 stdout
2 stderr

shell

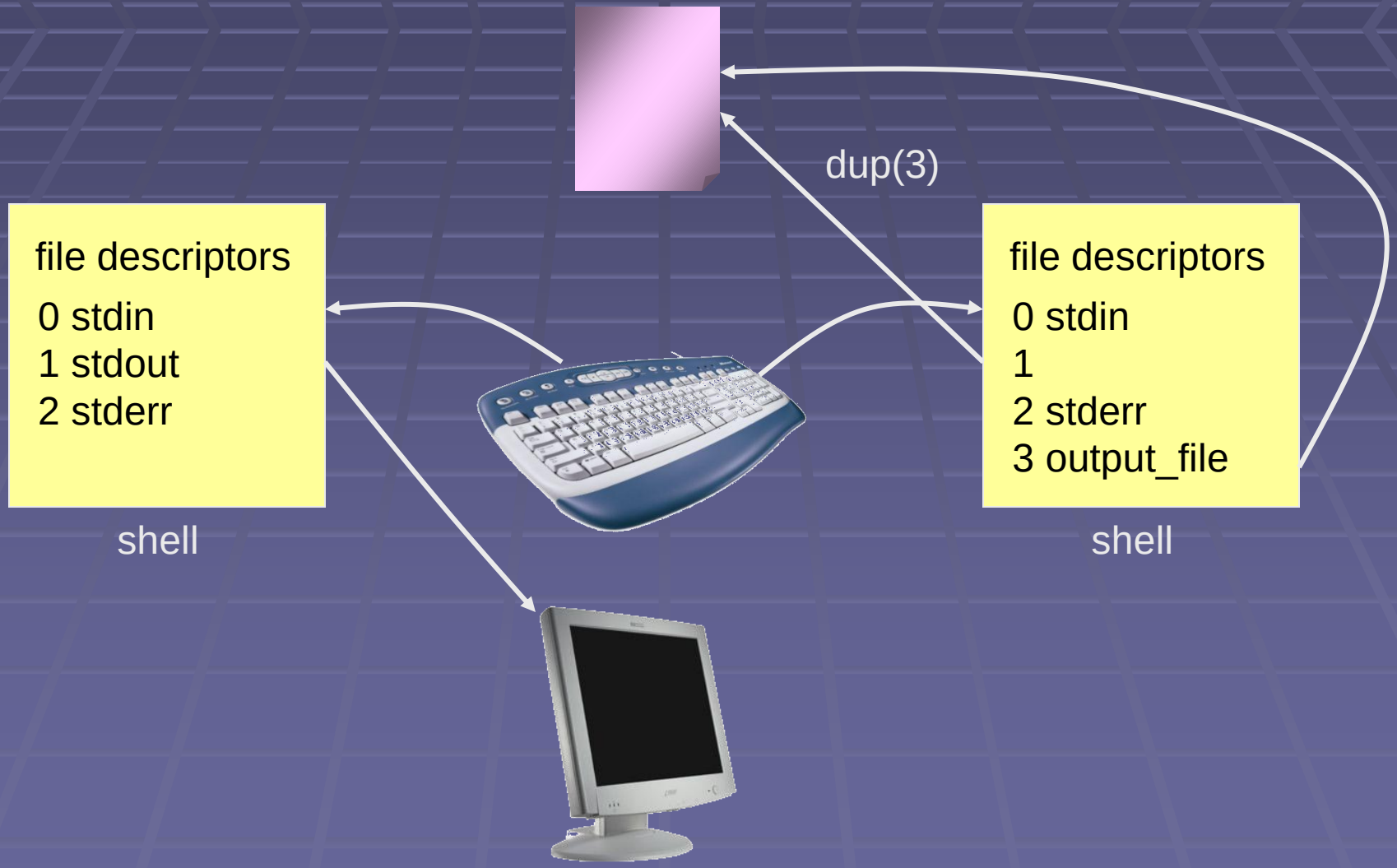
file descriptors
0 stdin
1 stdout
2 stderr
3 output_file

shell

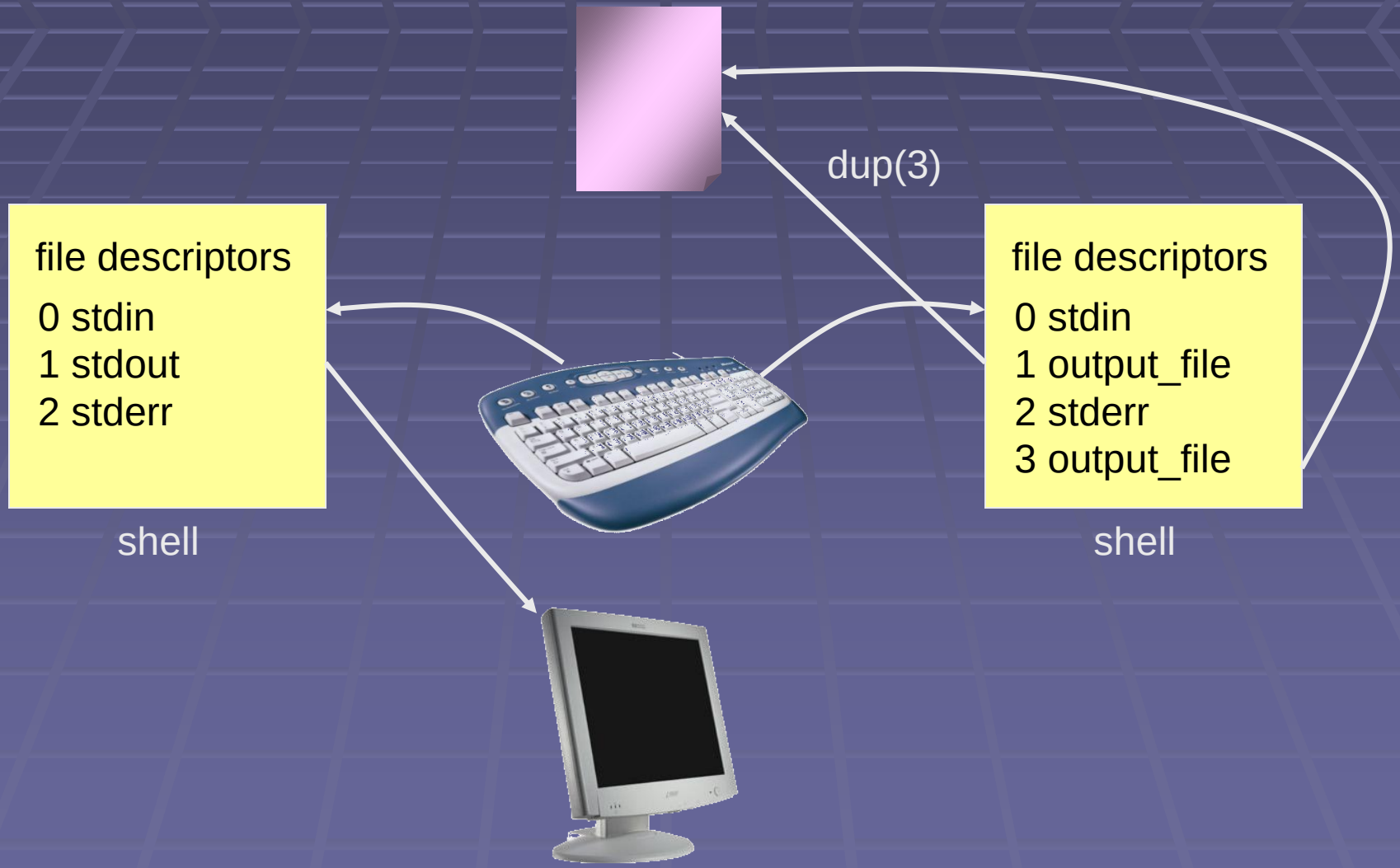
Output Redirection



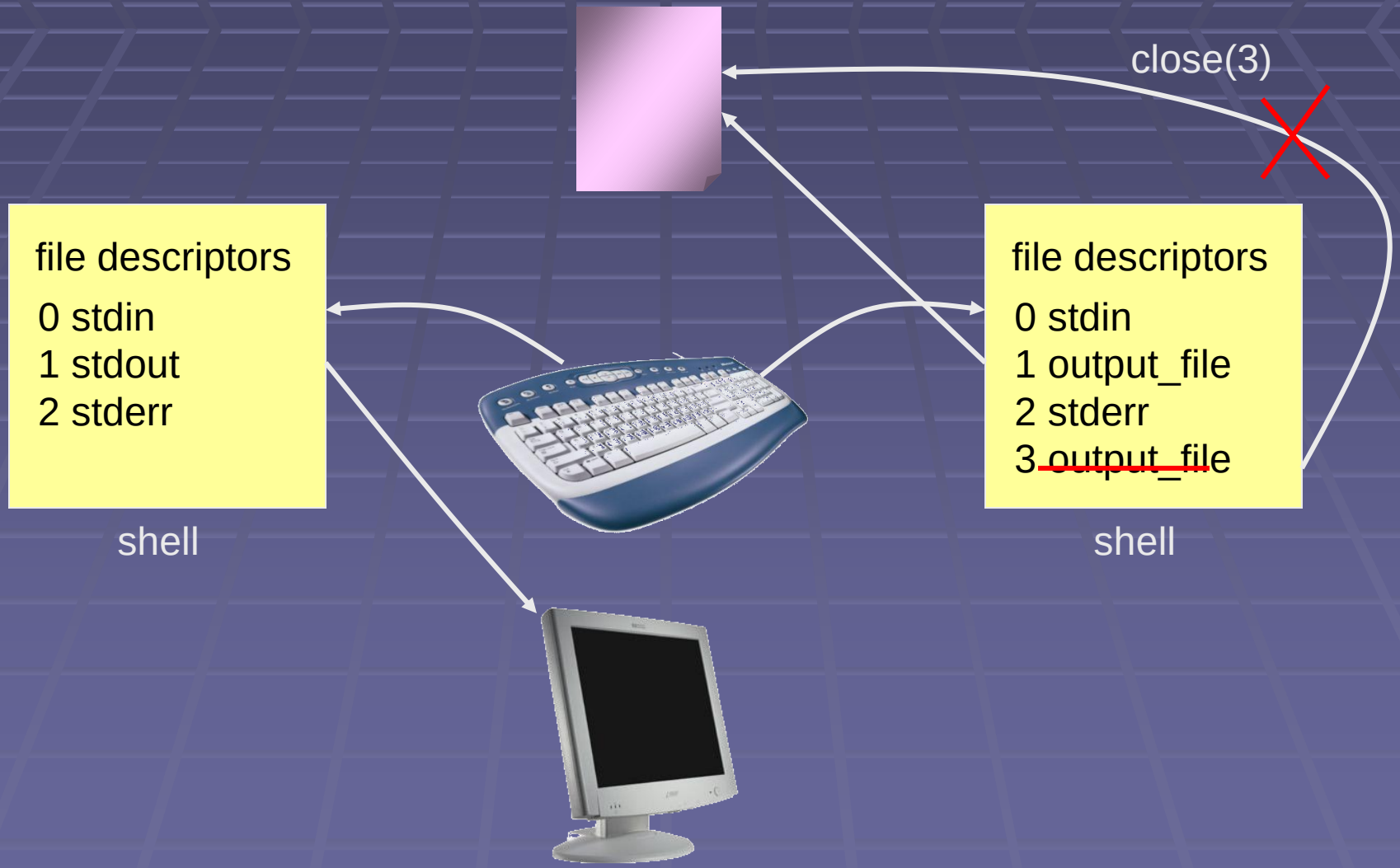
Output Redirection



Output Redirection



Output Redirection

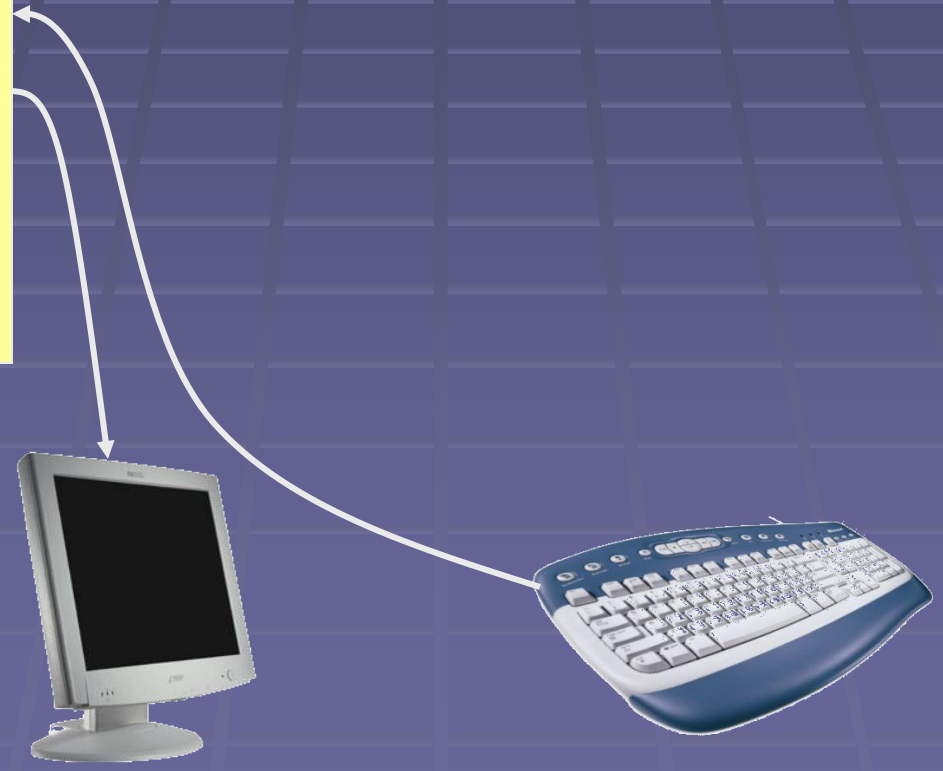


Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);
```

file descriptors
0 stdin
1 stdout
2 stderr

shell

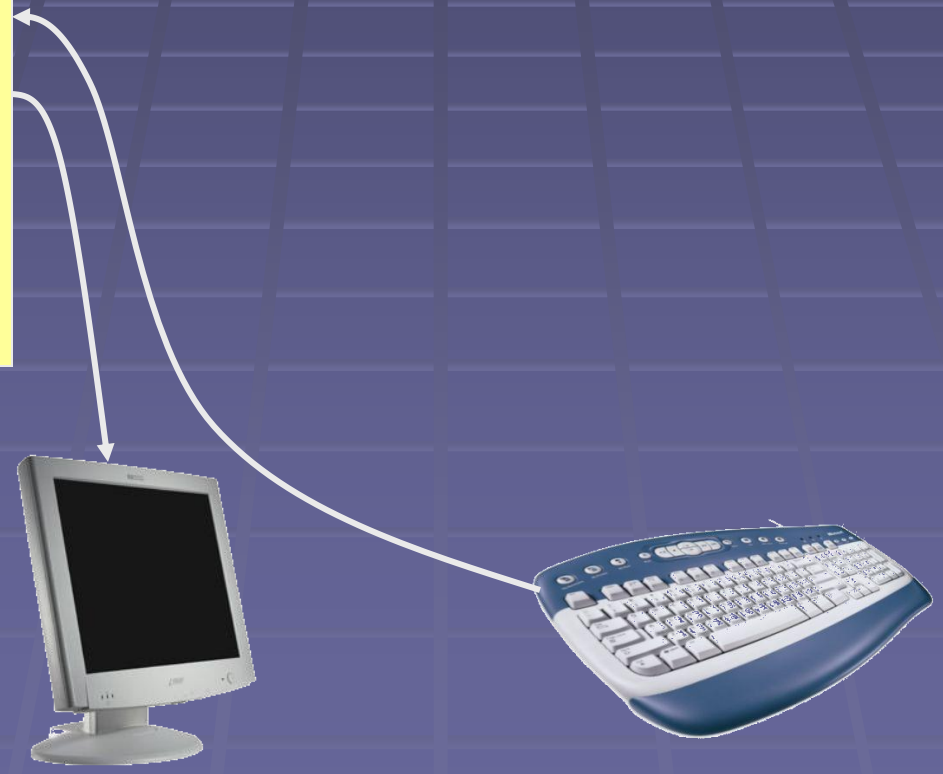


Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);
```

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

fork()

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

Pipe

fork()

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

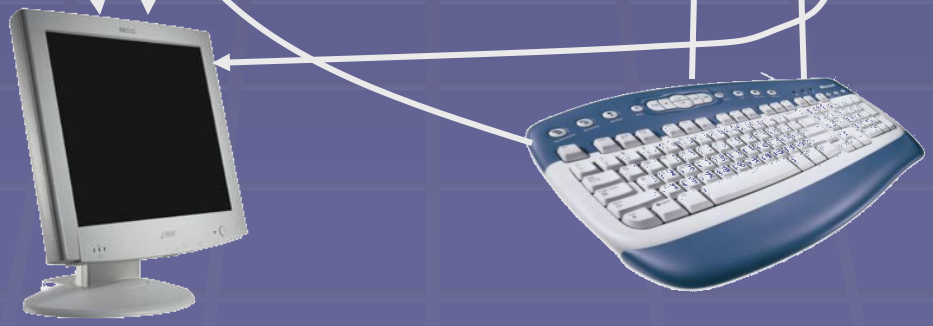
file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

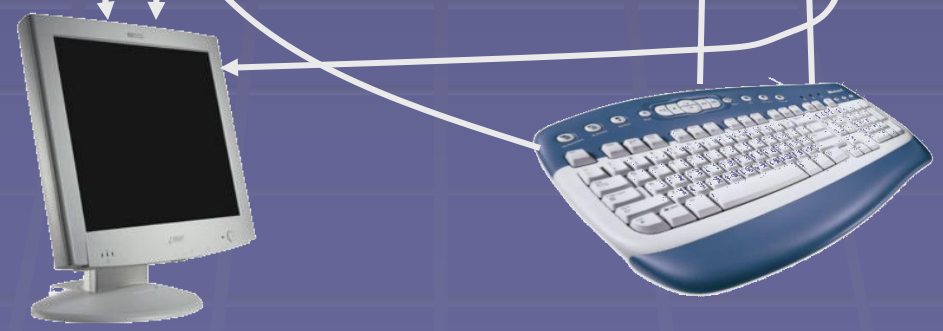
file descriptors
0 stdin
~~1 stdout~~
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

dup(4)

int p1_to_p2[2];
pipe(p1_to_p2);

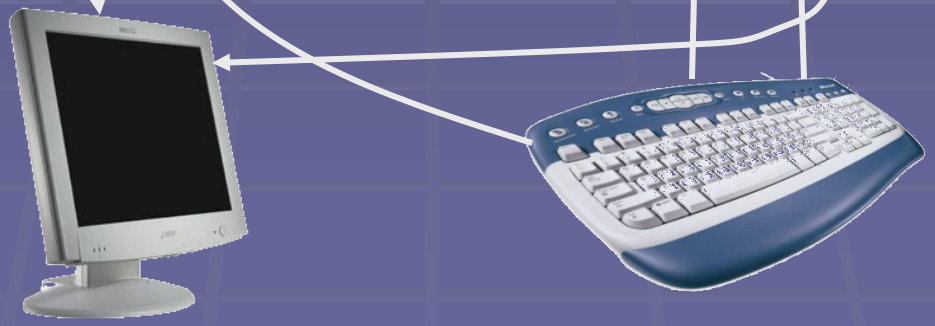
file descriptors
0 stdin
1
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

dup(4)

int p1_to_p2[2];
pipe(p1_to_p2);

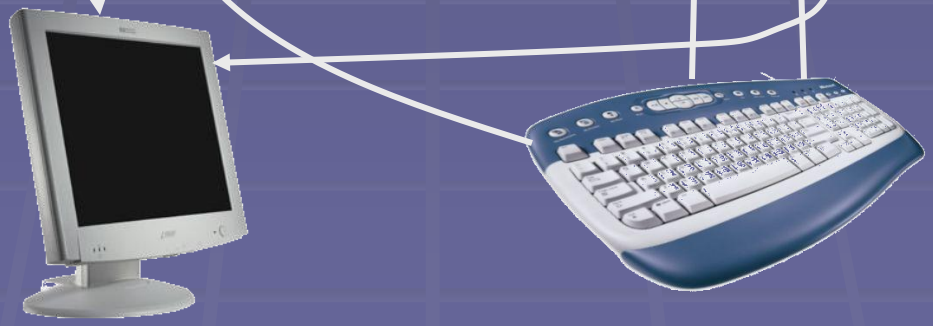
file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

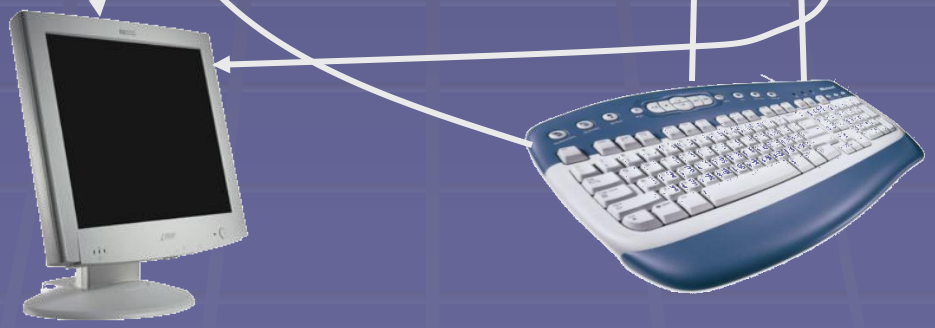
file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr
3 ~~p1_to_p2[0]~~
4 ~~p1_to_p2[1]~~

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

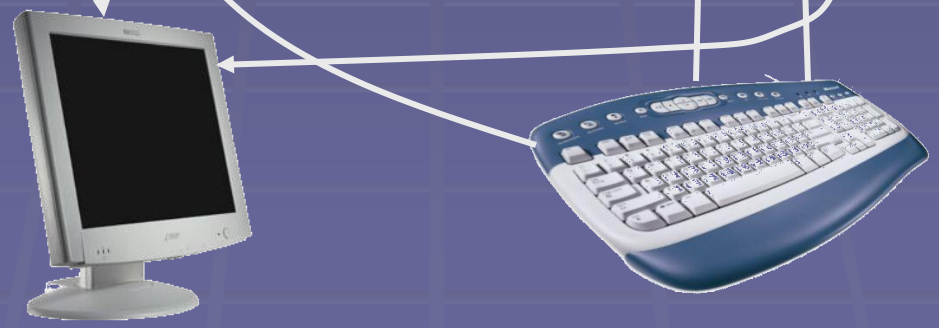
file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]
```

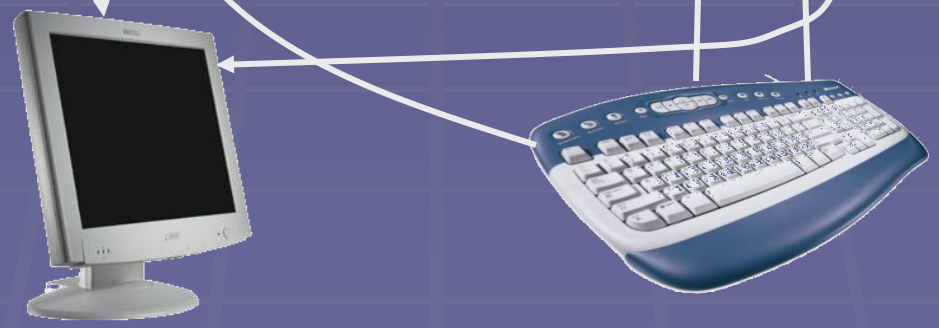
shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
  argv)
```

shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]
```

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

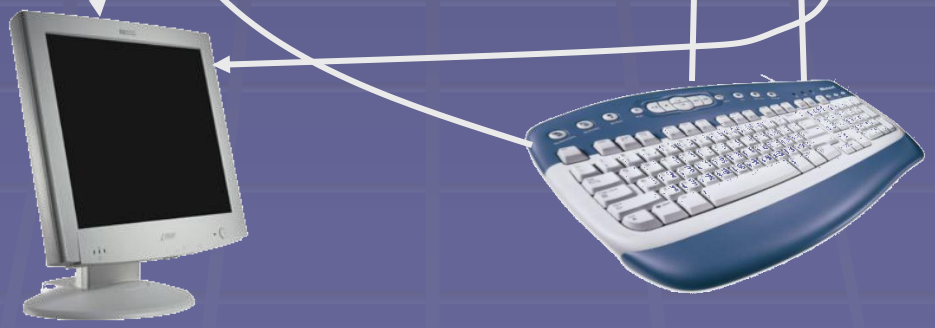
execvp(bin_path,
argv)

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 ~~stdin~~
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
argv)

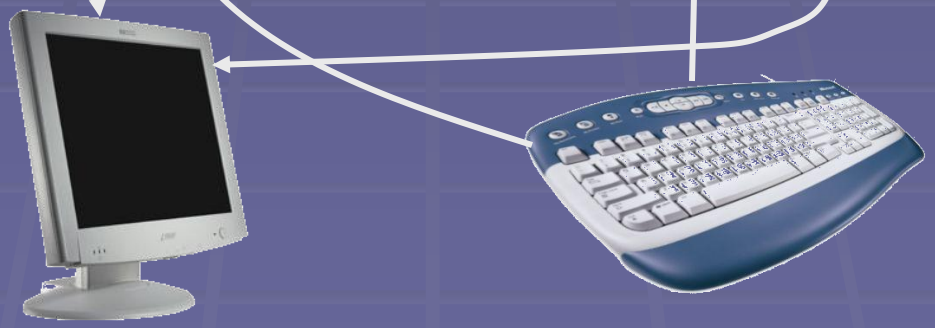
shell

dup(3)

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
argv)

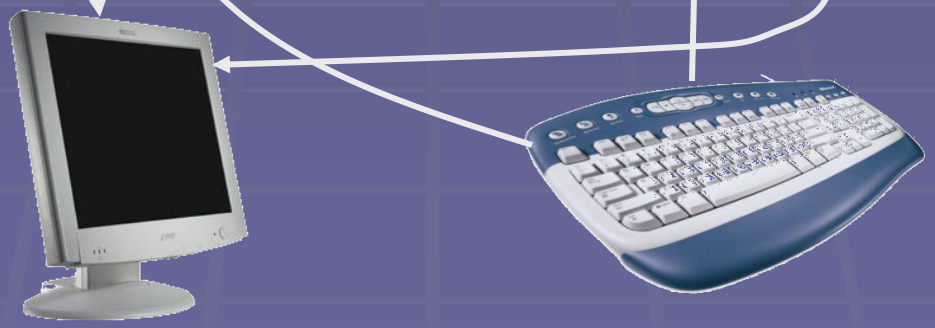
shell

dup(3)

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

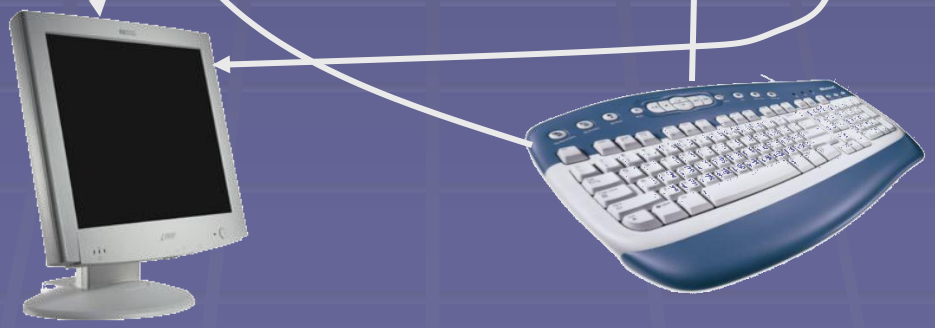
execvp(bin_path,
argv)

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr
3 ~~p1_to_p2[0]~~
4 ~~p1_to_p2[1]~~

shell



Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]
```

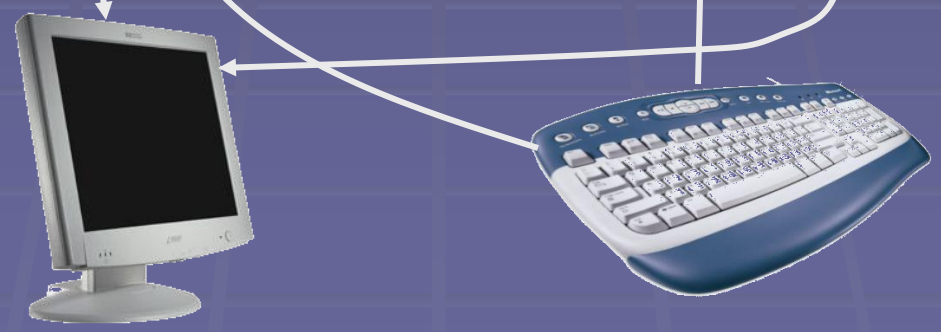
shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
  argv)
```

shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr
```

shell



Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 p1_to_p2[0]
4 p1_to_p2[1]
```

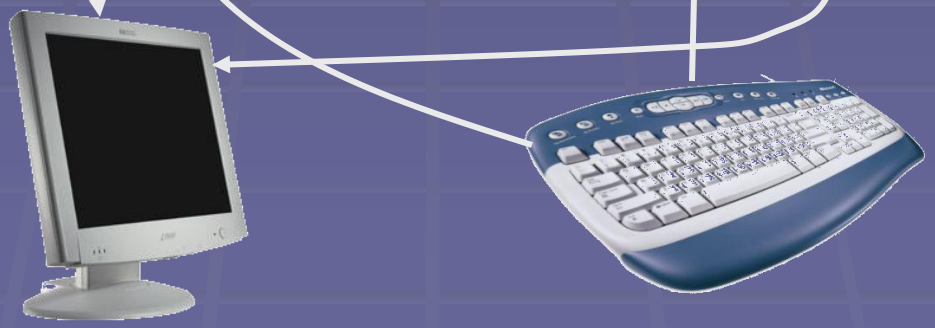
shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
  argv)
```

shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr

execvp(bin_path,
  argv)
```

shell



Pipe

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
3 ~~p1_to_p2[0]~~
4 ~~p1_to_p2[1]~~

shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
argv)

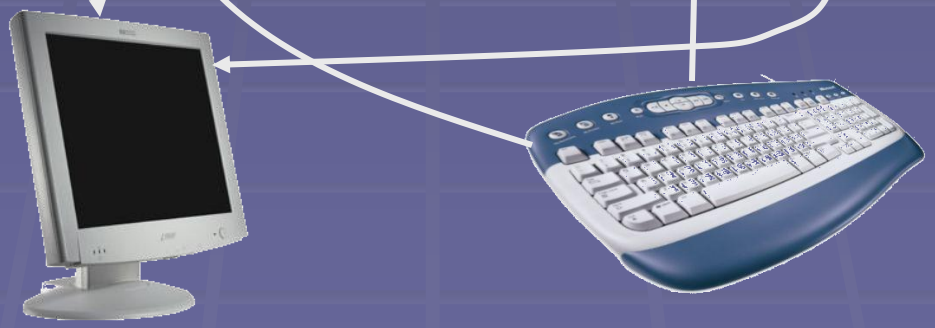
shell

int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr

execvp(bin_path,
argv)

shell



Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr
```

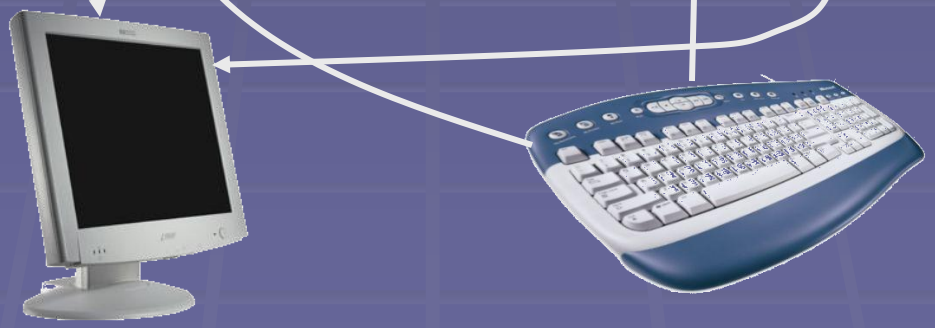
shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
  argv)
```

shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr

execvp(bin_path,
  argv)
```

shell



Pipe

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 stdout
2 stderr

waitpid(…)
waitpid(…)
```

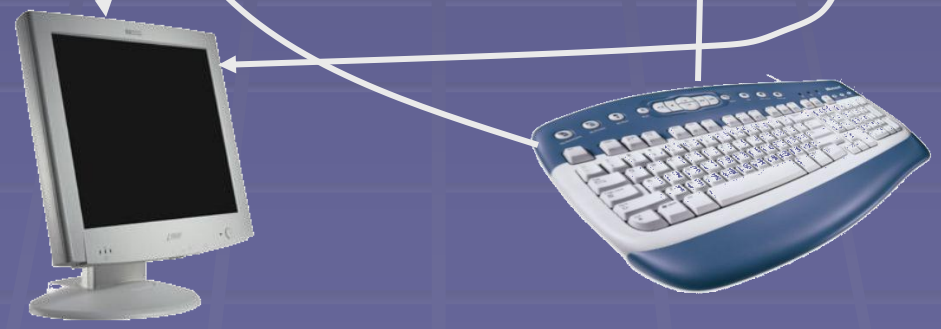
shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 stdin
1 p1_to_p2[1]
2 stderr

execvp(bin_path,
  argv)
```

shell

```
int p1_to_p2[2];
pipe(p1_to_p2);

file descriptors
0 p1_to_p2[0]
1 stdout
2 stderr

execvp(bin_path,
  argv)
```

shell



Child Process

- Once a process exits some information may be retained
  - Allow parent to determine exit status
- wait*()
  - Check the exit status
- zombie
  - Child terminated but wait not called