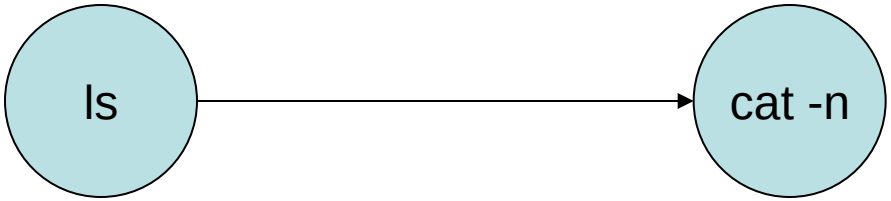
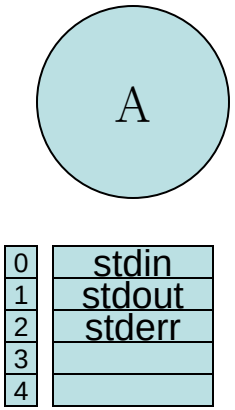
Pipe

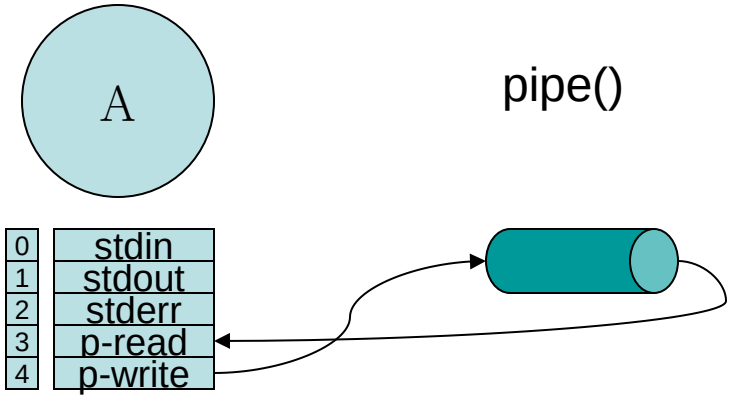
Pipe-basic



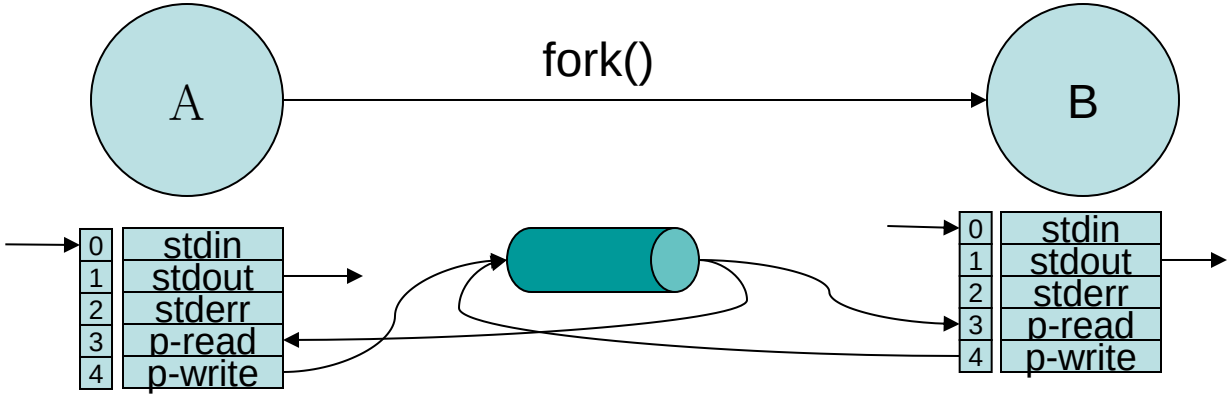
Pipe-basic



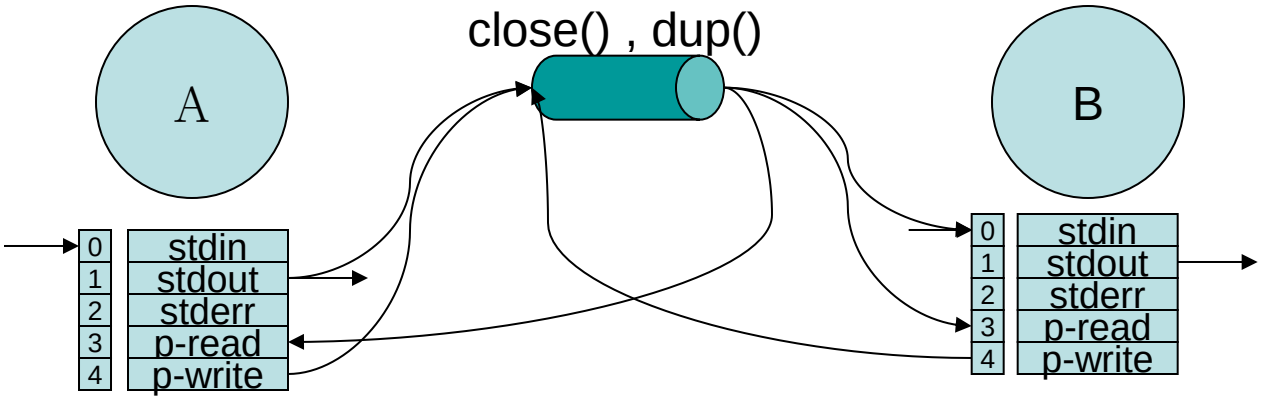
Pipe-basic



Pipe-basic


A
fork()
B
0
1
2
3
4
stdin
stdout
stderr
p-read
p-write
0
1
2
3
4
stdin
stdout
stderr
p-read
p-write

Pipe-basic



close(1)

dup(4)

close(3)

close(4)

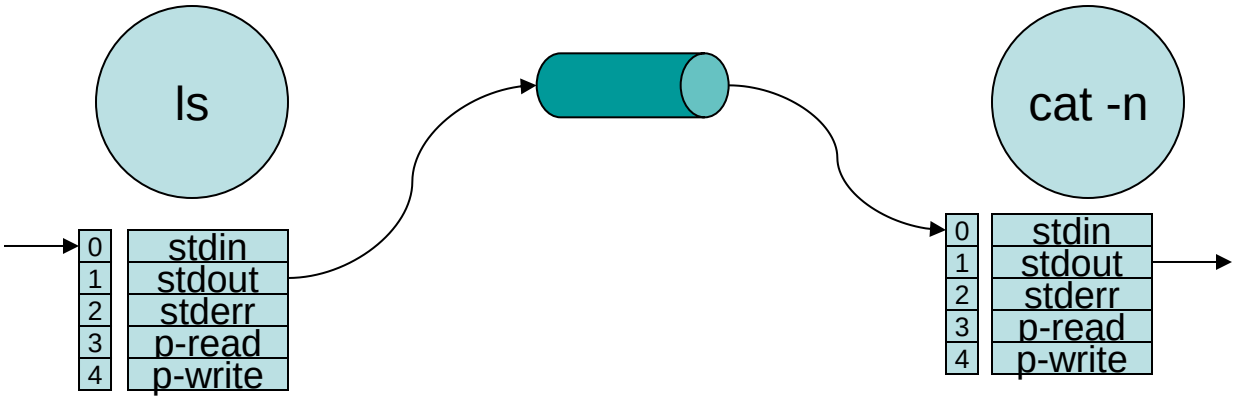
close(0)

dup(3)

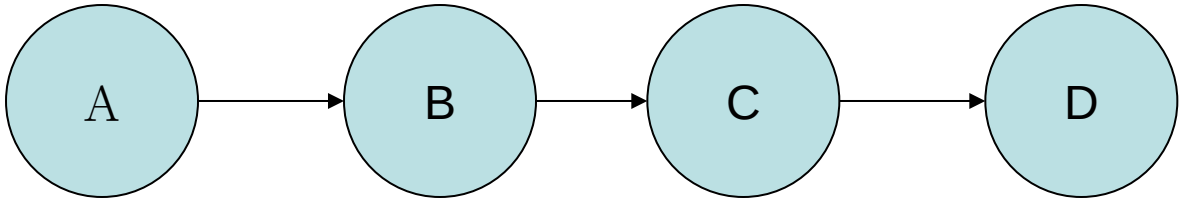
close(3)

close(4)

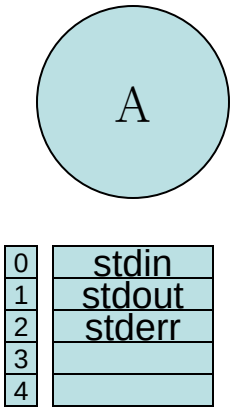
Pipe-basic


ls
cat -n
0
1
2
3
4
stdin
stdout
stderr
p-read
p-write
0
1
2
3
4
stdin
stdout
stderr
p-read
p-write

One-to-one pipe model

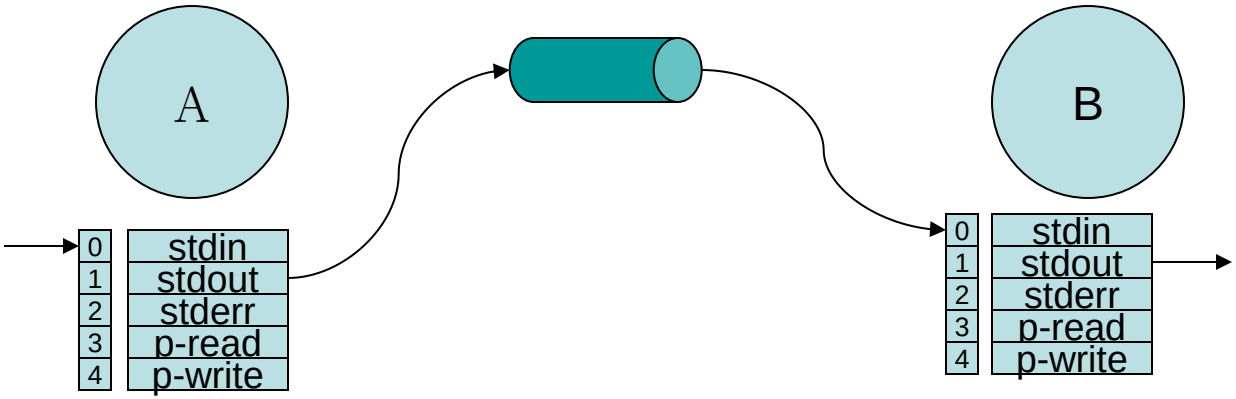


One-to-one pipe model

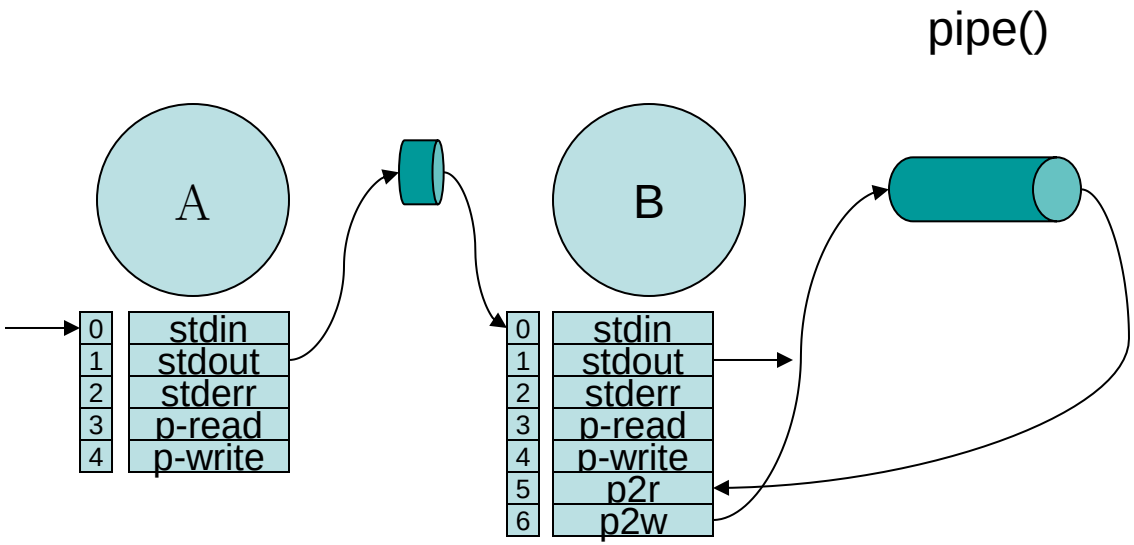


One-to-one pipe model

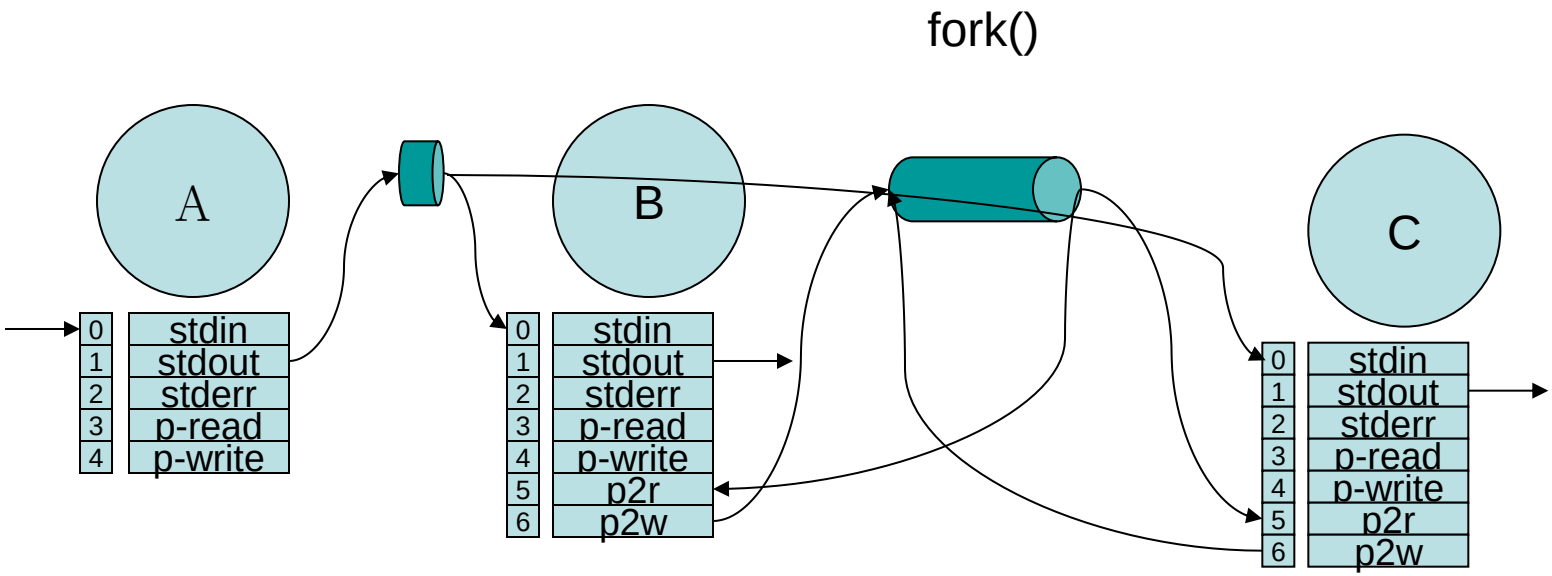
PIPE() // A → B



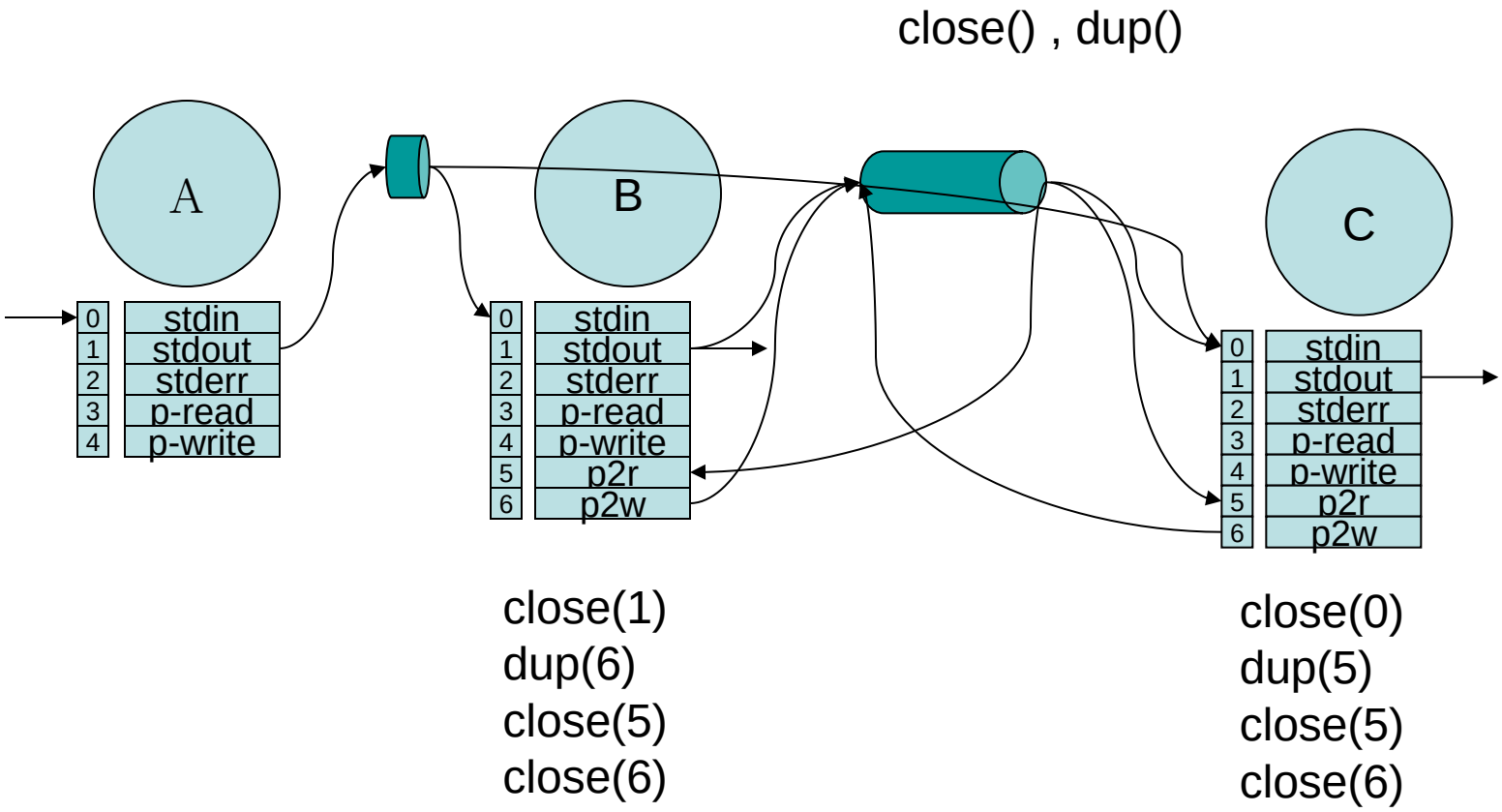
One-to-one pipe model



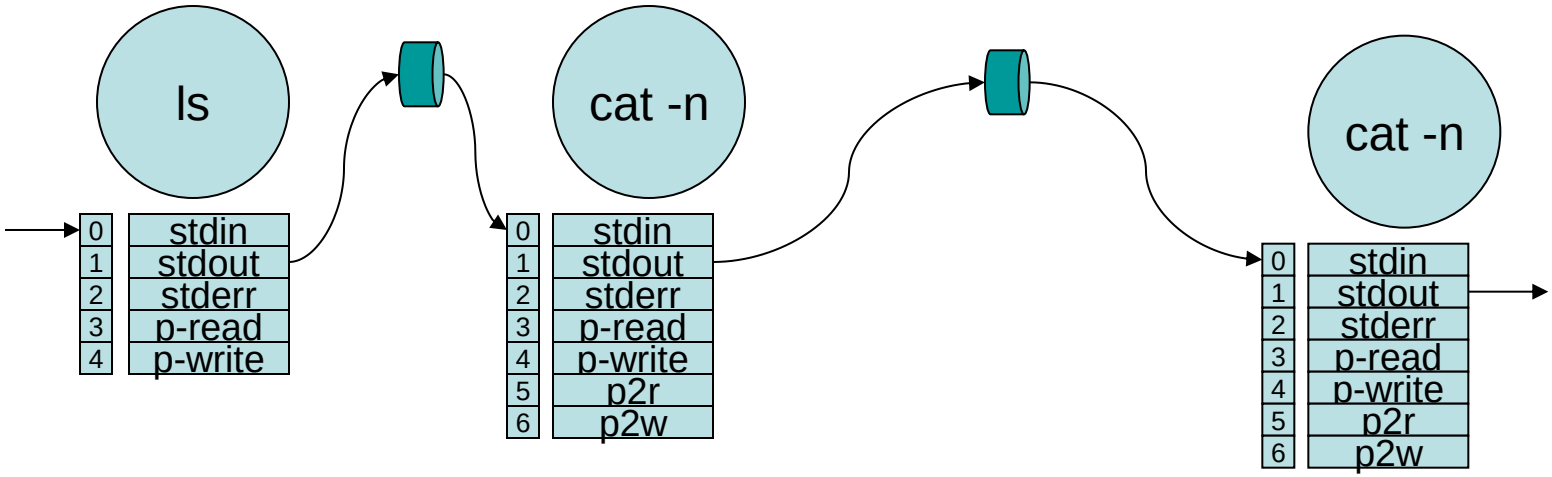
One-to-one pipe model



One-to-one pipe model



One-to-one pipe model



Useful functions

#include <unistd.h>

Name	Description
pid_t fork(void)	create a new process by duplicating the calling process.

Name
int execl(const char *_path_, const char *_arg_, ...) int execlp(const char *_file_, const char *_arg_, ...) int execle(const char *_path_, const char *_arg_ , ..., char * const _envp_[]) int execv(const char *_path_, char *const _argv_[]) int execvp(const char *_file_, char *const _argv_[])
Description
replace the current process image with a new process image. The first argument should point to the file name associated with the file being executed. arg or argv are argument list available to the executed program.

Useful functions

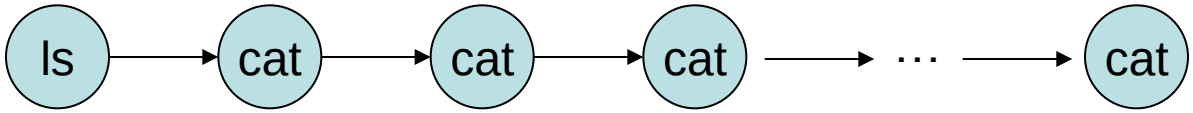
#include <unistd.h>

Name	Description
int pipe (*int filedes*[2])	create a pipe and put the file descriptors for the reading and writing ends of the pipe (respectively) into *filedes*[0] and *filedes*[1].
int dup(int *oldfd*)	create a copy of the file descriptor *oldfd* and uses the lowest-numbered unused descriptor for the new descriptor.
int dup2(int *oldfd*, int *newfd*);	create a copy of the file descriptor *oldfd* and make *newfd* be the copy of *oldfd*, closing *newfd* first if necessary.
int close(int *fd*)	close a file descriptor, so that it no longer refers to any file and may be reused.

Note

- Close fd cleanly. Before reading the pipe, write should be closed.

Special case



> ls | cat | cat | cat | …

Special case

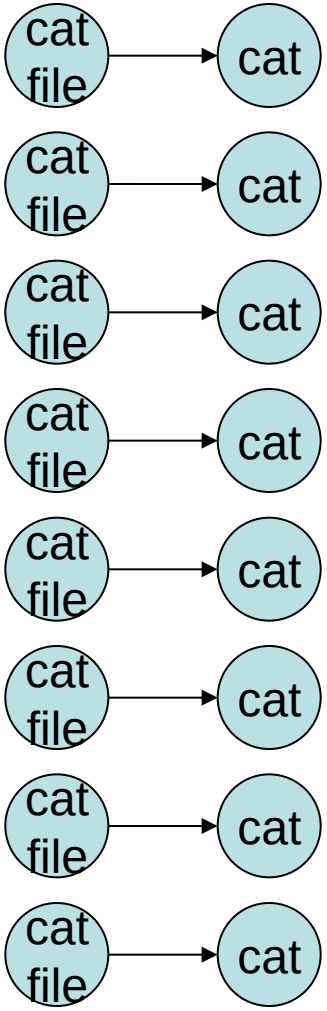
> cat file |500

> cat file |500

> cat file |500

……

> cat

> cat

> cat