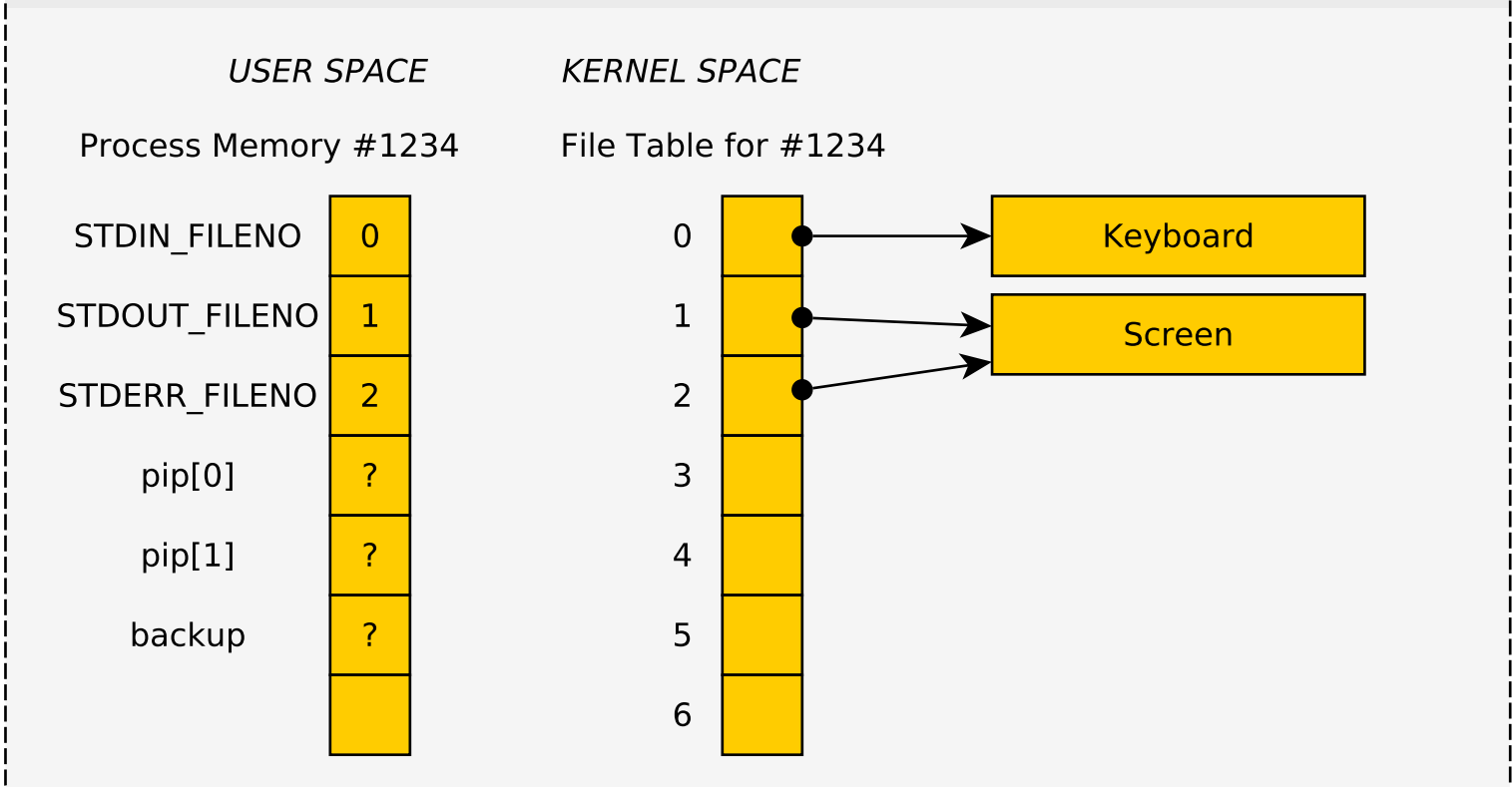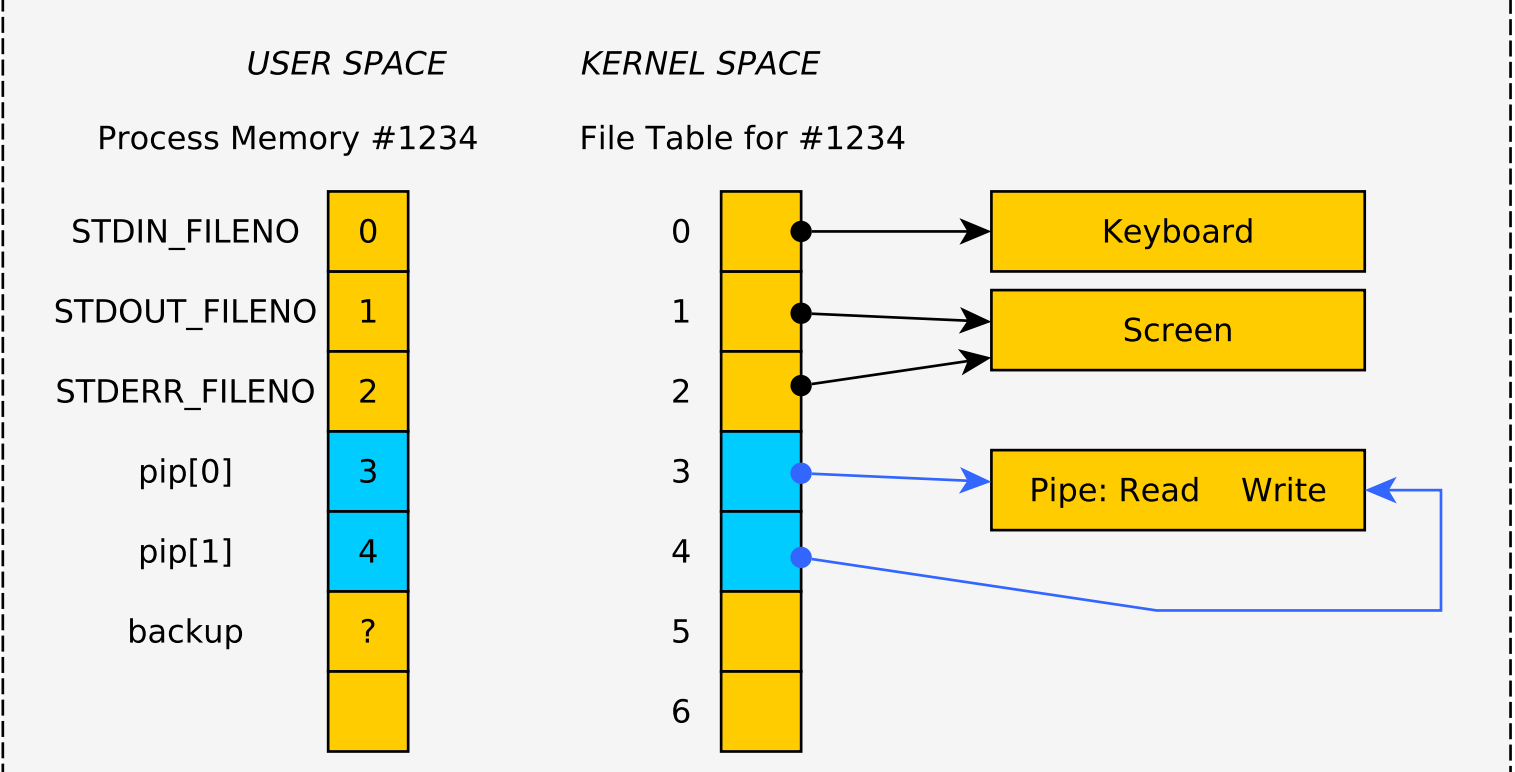This set of diagrams shows how calls to pipe(), dup(), and dup2() can be used to redirect output that would go to the screen into a pipe. Each grouping is preceded by a system call which changes the state to what appears in the diagram below.
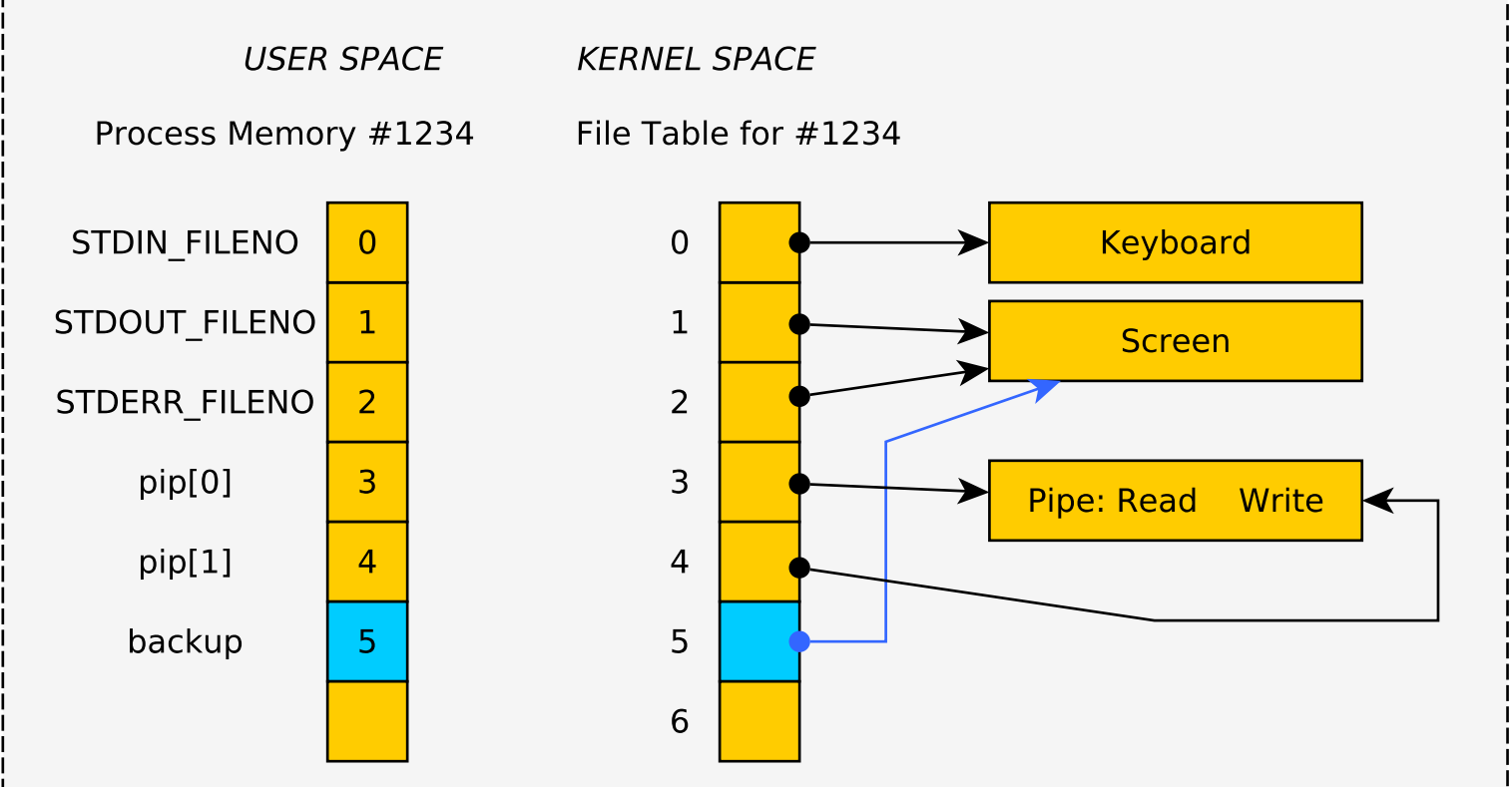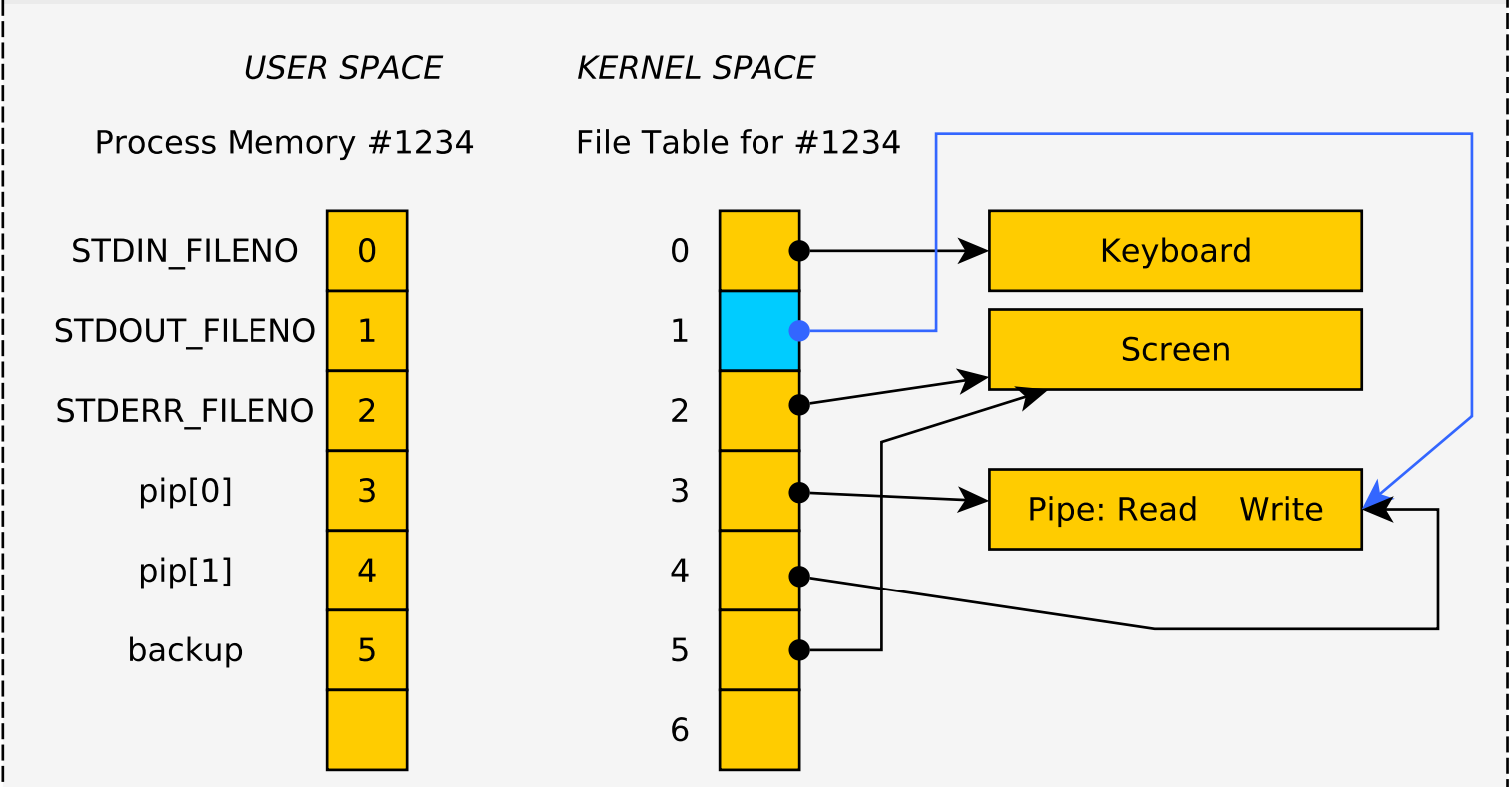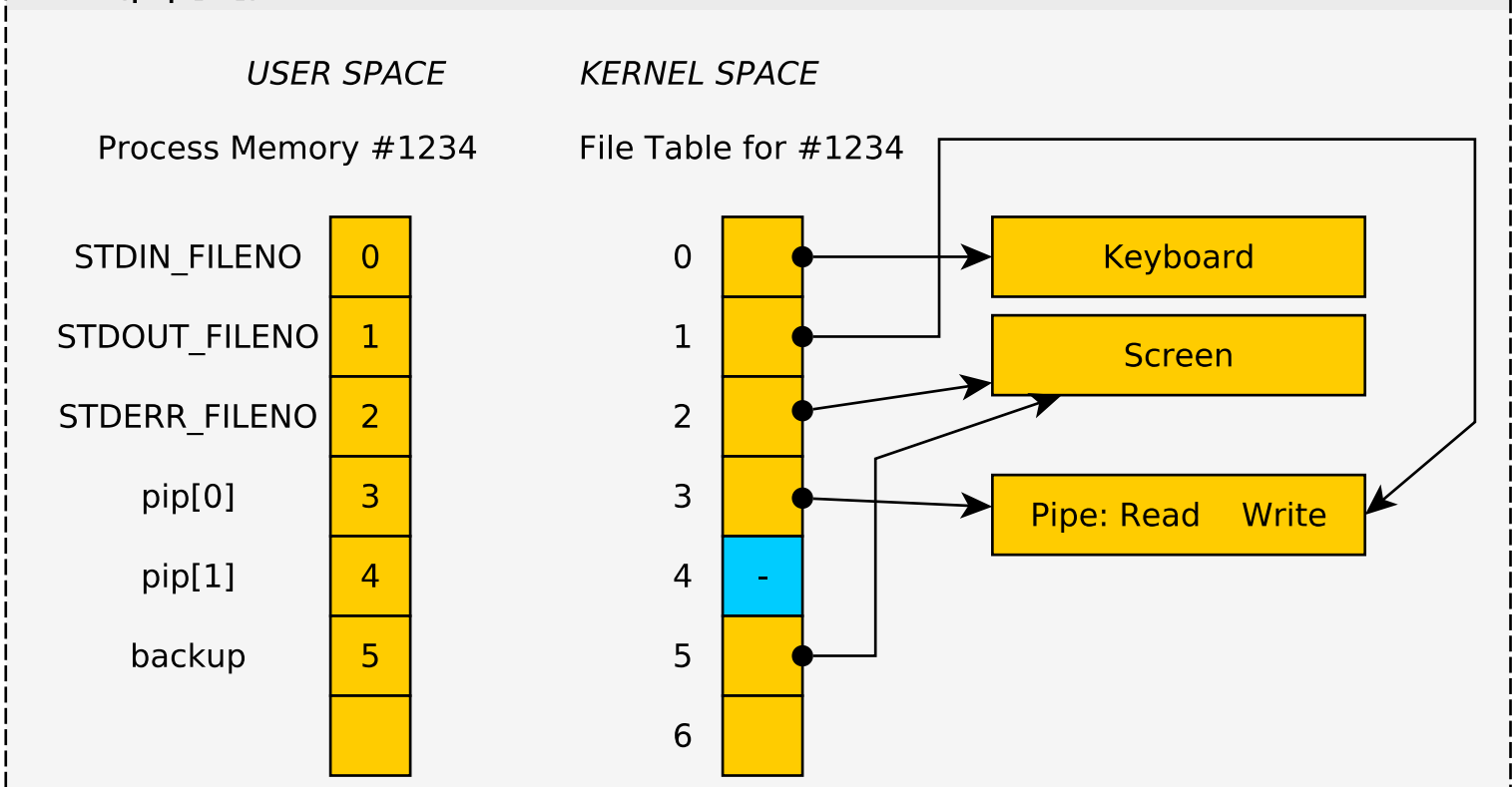
Initial

USER SPACE — Process Memory #1234

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	?
pip[1]	?
backup	?

KERNEL SPACE — File Table for #1234: 0 → Keyboard; 1 → Screen; 2 → Screen

pipe(pip);

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	3
pip[1]	4
backup	?

Pipe: Read   Write

backup = dup(STDOUT_FILENO);

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	3
pip[1]	4
backup	5

dup2(pip[1], STDOUT_FILENO);

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	3
pip[1]	4
backup	5

close(pip[1]);

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	3
pip[1]	4 → -
backup	5

Any printing that is done with printf() or other references to stdout or STDOUT_FILENO will print into the write end of the pipe currently. The next step restores standard out so output goes to the screen again.

dup2(backup, STDOUT_FILENO);

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
pip[0]	3
pip[1]	4 → -
backup	5