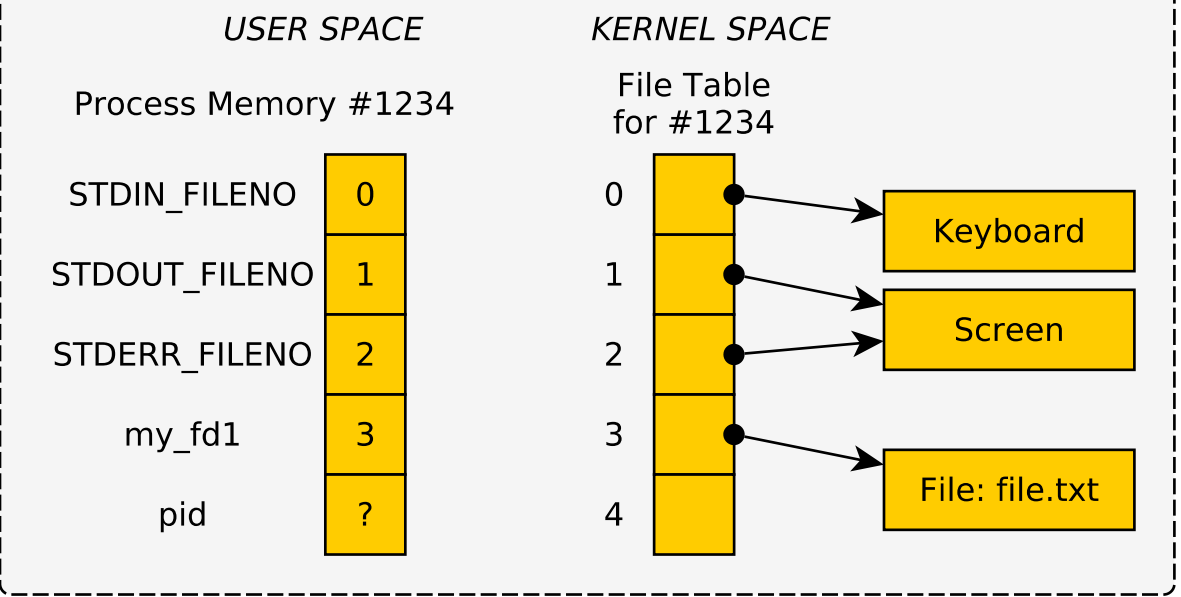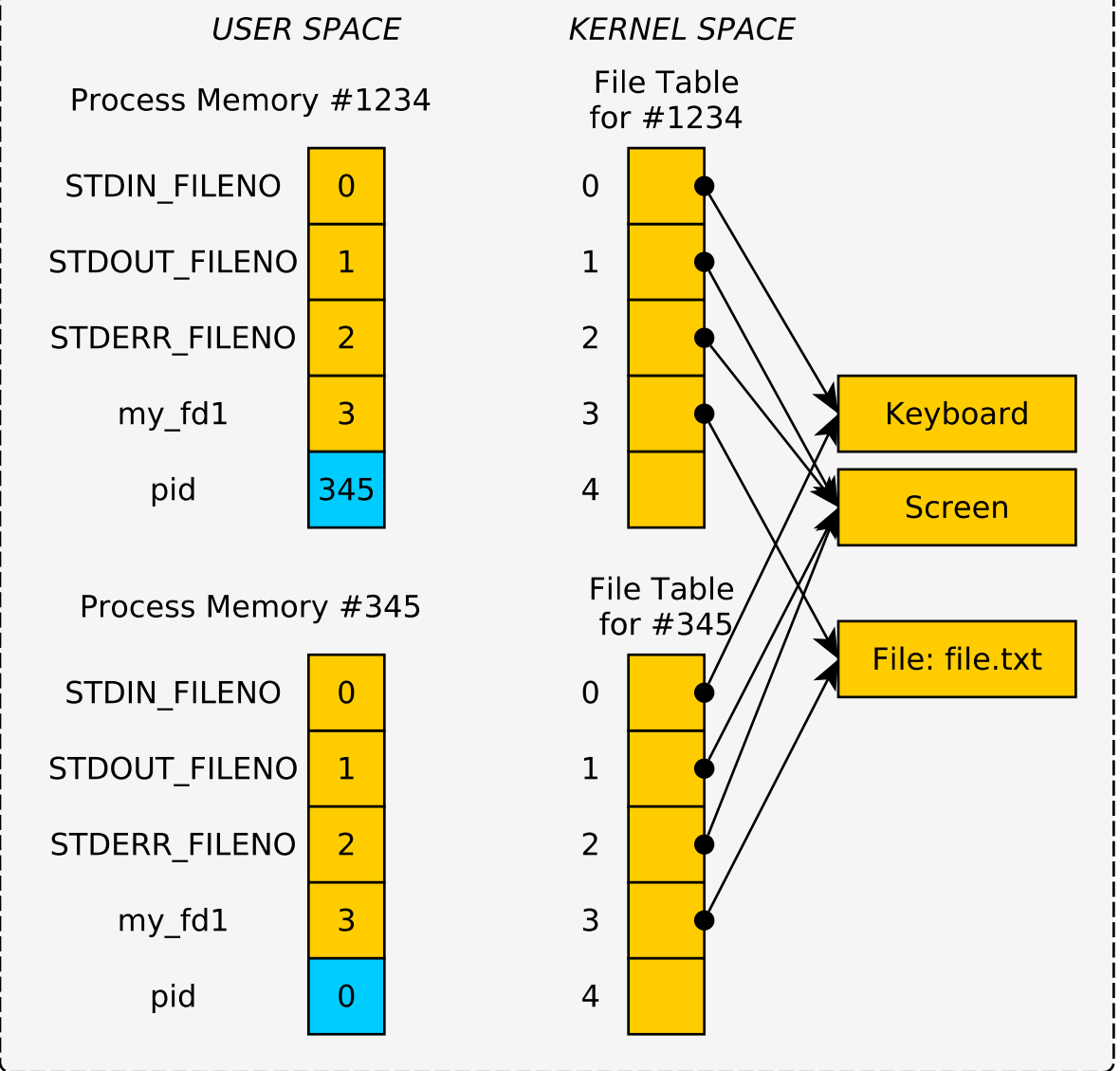This diagram shows a typical sequence in which a parent process opens a file, forks(), then has the child redirect standard output to that file.
It is similar to what shells do to redirect output in commands like
  ls -l > output.txt
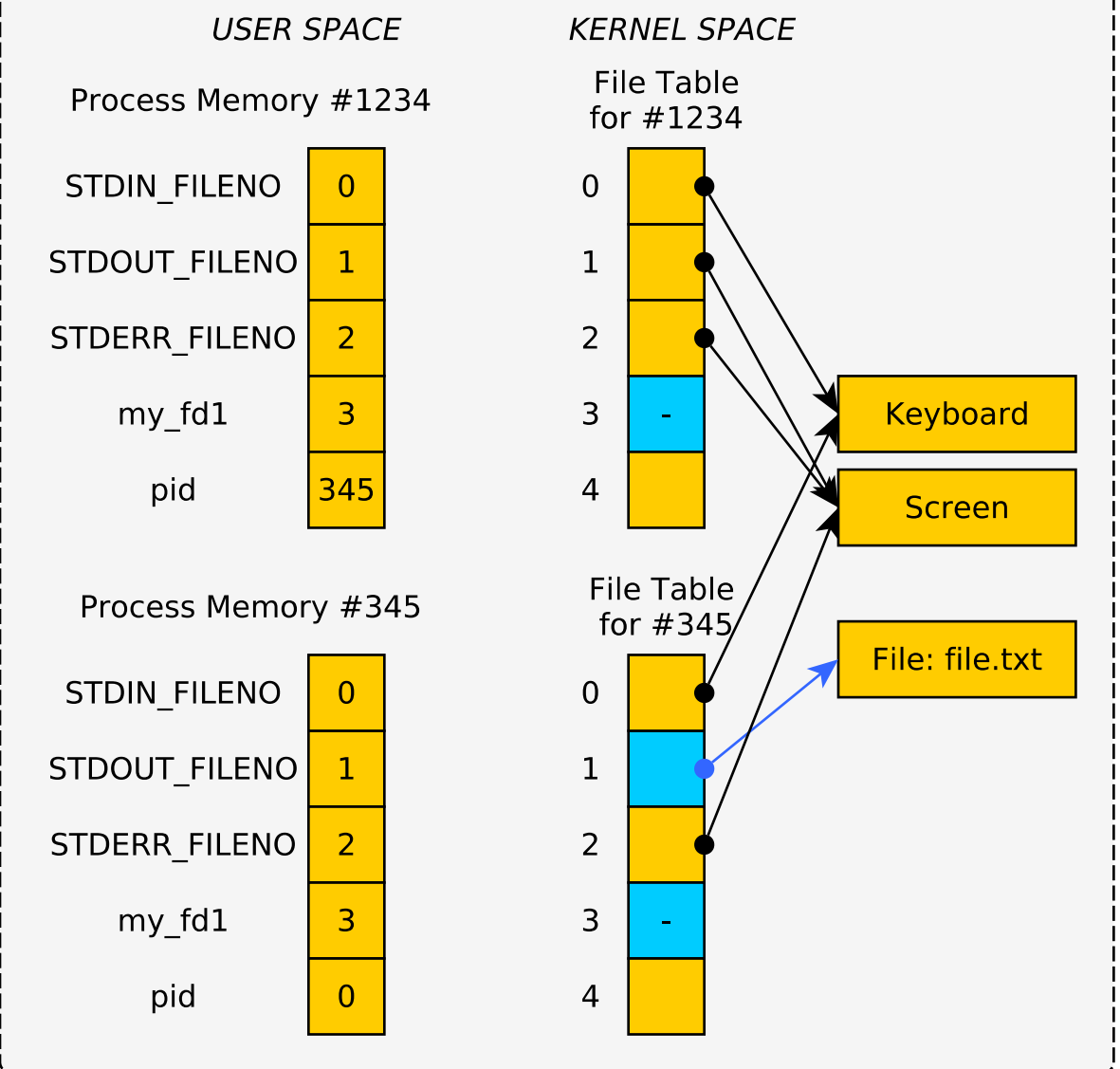which will put the output listing the file output.txt.

Initial

USER SPACE | KERNEL SPACE

Process Memory #1234

File Table for #1234

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
my_fd1	3
pid	?

0 → Keyboard
1 → Screen
2 → Screen
3 → File: file.txt
4

pid = fork();

USER SPACE | KERNEL SPACE

Process Memory #1234

File Table for #1234

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
my_fd1	3
pid	345

Process Memory #345

File Table for #345

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
my_fd1	3
pid	0

Keyboard
Screen
File: file.txt

CHILD: dup2(my_fd1, STDOUT_FILENO);
BOTH: close(my_fd1);

USER SPACE | KERNEL SPACE

Process Memory #1234

File Table for #1234

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
my_fd1	3
pid	345

Process Memory #345

File Table for #345

STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2
my_fd1	3
pid	0

Keyboard
Screen
File: file.txt

Subsequently, if the child process calls printf(), output will go into file.txt. Calling exec() will cause whatever program takes the place of the process to write to the file rather than to the screen.