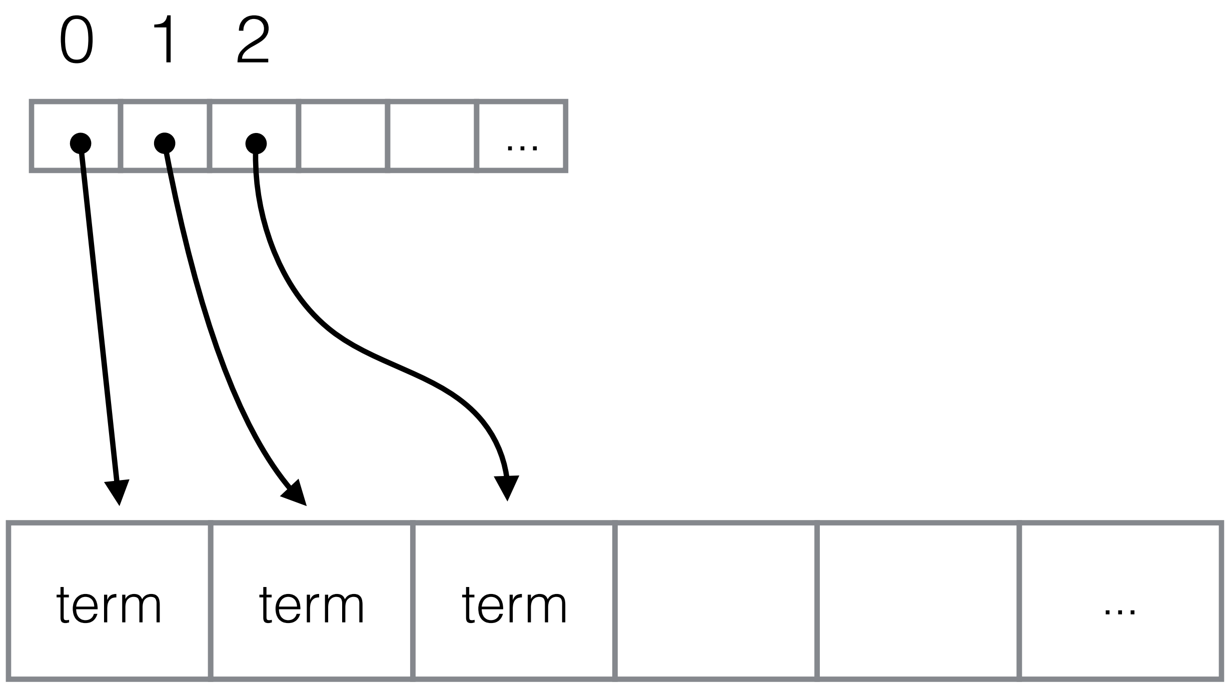
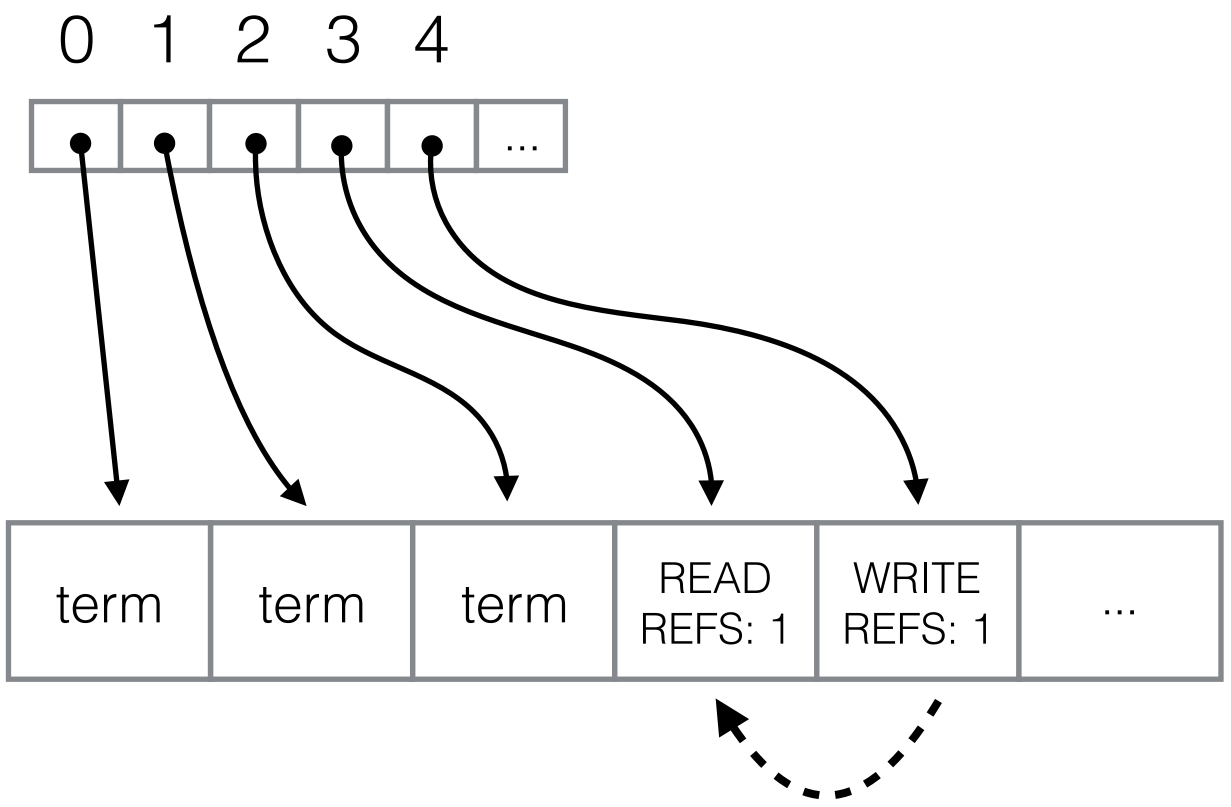
subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
```



subprocess

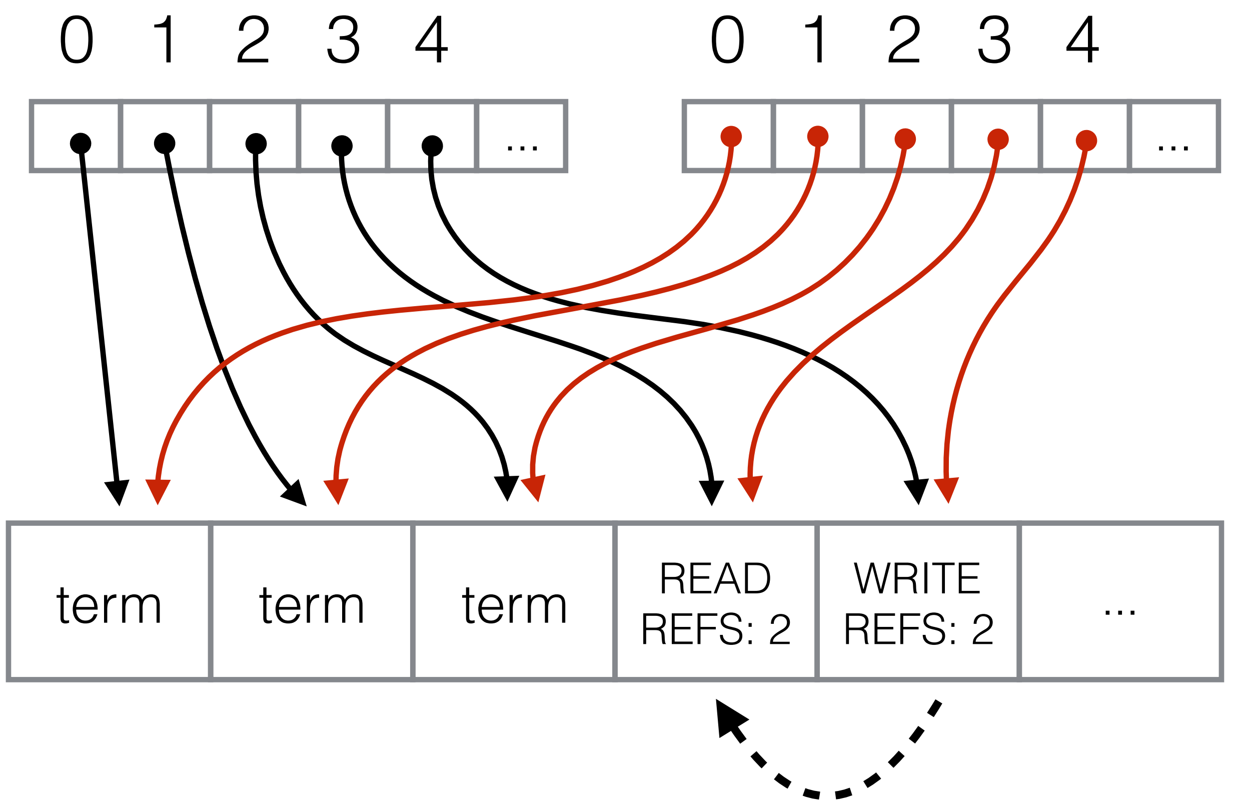
```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
```



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
```

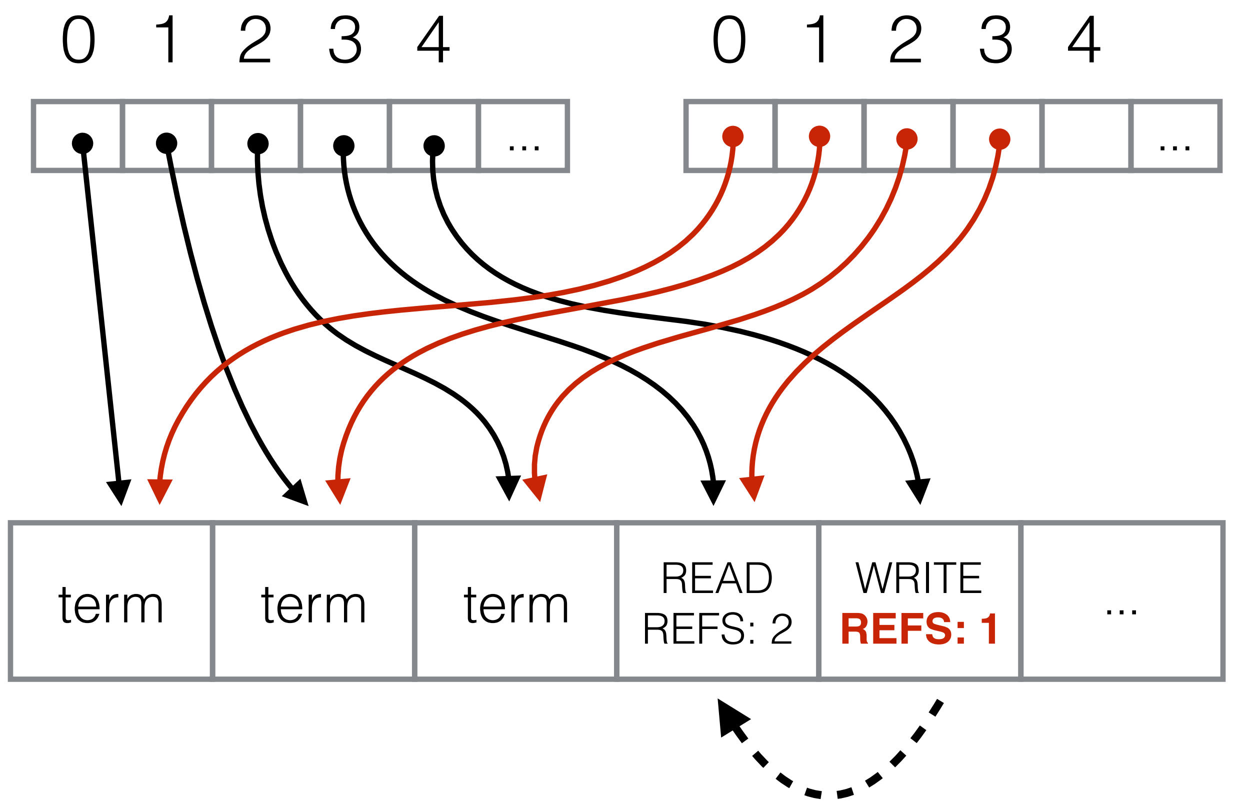
`process.infd: 4`



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
    if (process.pid == 0) {
        close(fds[1]);
```

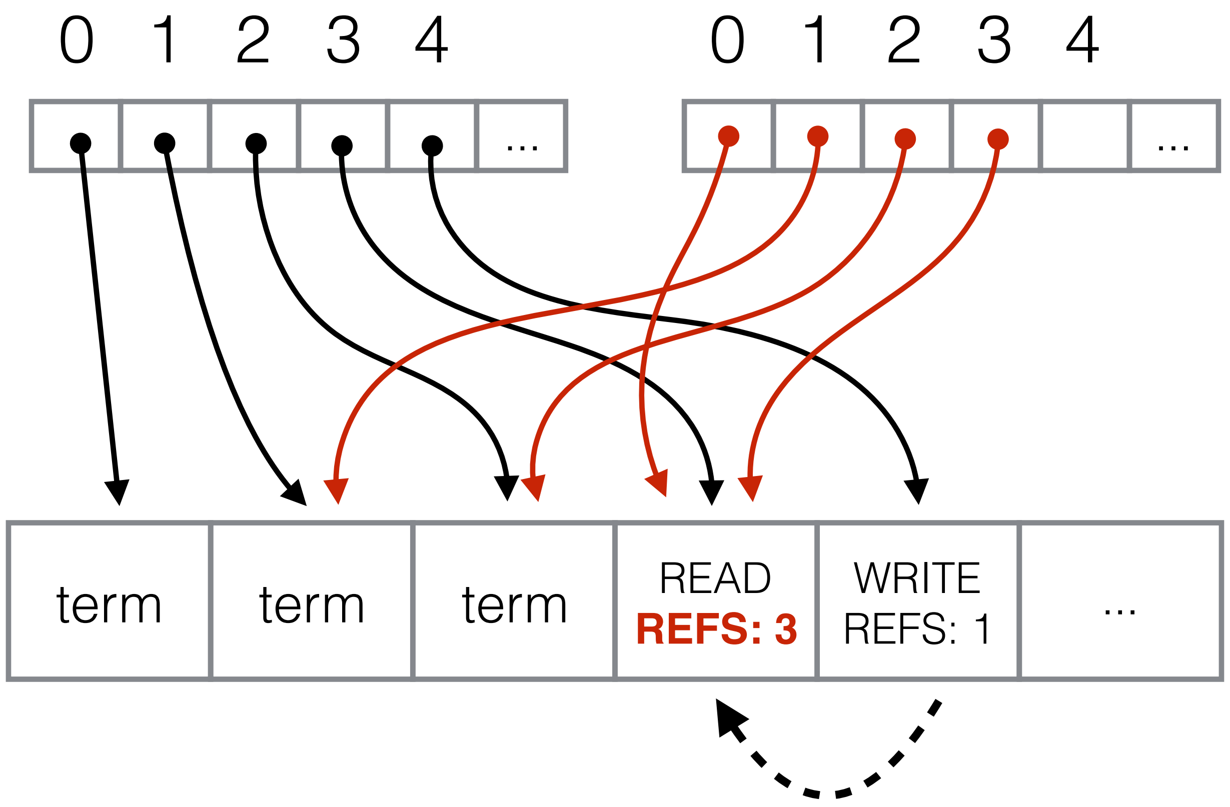
`process.infd: 4`



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
    if (process.pid == 0) {
        close(fds[1]);
        dup2(fds[0], STDIN_FILENO);
```

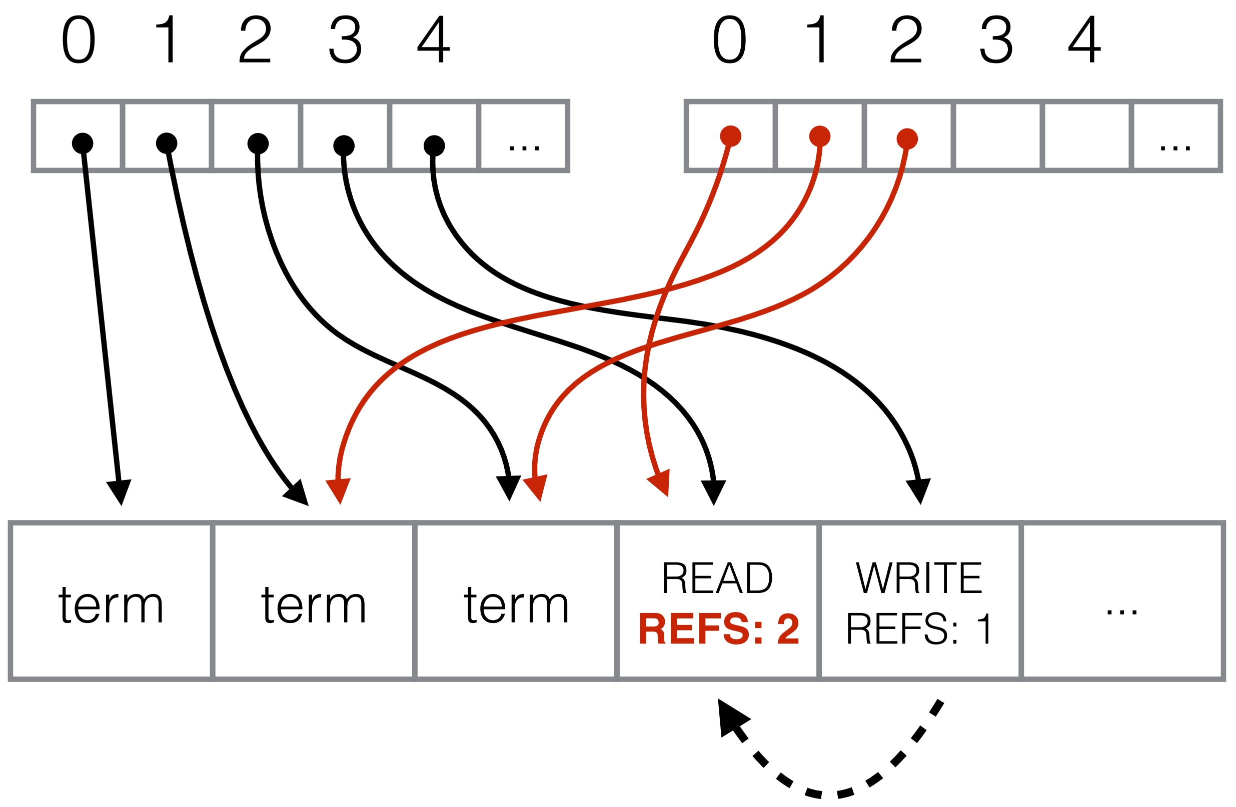
```
process.infd: 4
```



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
    if (process.pid == 0) {
        close(fds[1]);
        dup2(fds[0], STDIN_FILENO);
        close(fds[0]);

        char *argv[] = {...};
        execvp(argv[0], argv);
    }
```

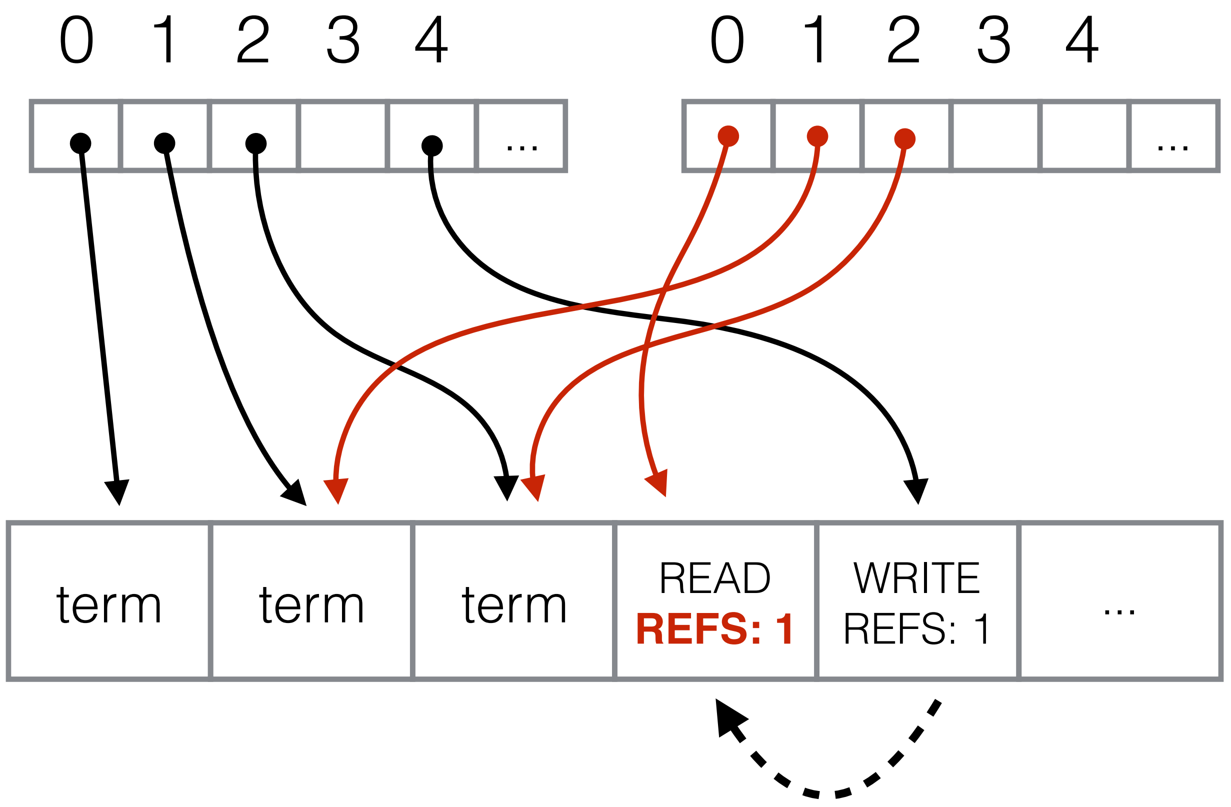
```
process.infd: 4
```



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
    if (process.pid == 0) {
        close(fds[1]);
        dup2(fds[0], STDIN_FILENO);
        close(fds[0]);

        char *argv[] = {...};
        execvp(argv[0], argv);
    }

    // Parent:
    close(fds[0]);

    return process;
}
```

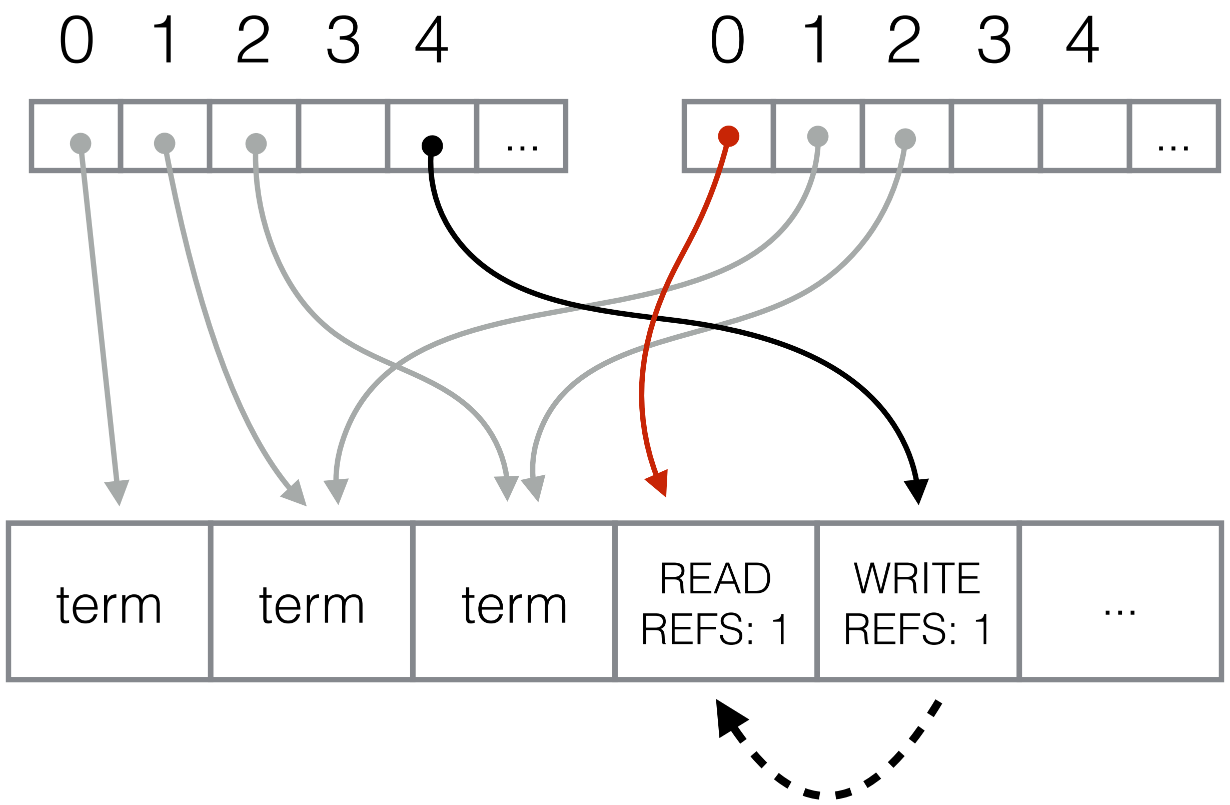
```
process.infd: 4
```



subprocess

```
subprocess_t subprocess(const char *command) {
    int fds[2];
    pipe(fds);
    subprocess_t process = { fork(), fds[1] };
    if (process.pid == 0) {
        close(fds[1]);
        dup2(fds[0], STDIN_FILENO);
        close(fds[0]);

        char *argv[] = {...};
        execvp(argv[0], argv);
    }

    // Parent:
    close(fds[0]);

    return process;
}

int main(int argc, char *argv[]) {
    subprocess_t sp = subprocess("/usr/bin/sort");
    const char *words[] = {"pen", "pineapple", "apple", "pen"};
    for (size_t i = 0; i < sizeof(words) / sizeof(words[0]); i++) {
        dprintf(sp.infd, "%s\n", words[i]);
    }
    close(sp.infd);

    int status;
    pid_t pid = waitpid(sp.pid, &status, 0);
    return pid == sp.pid && WIFEXITED(status) ? WEXITSTATUS(status) : -1;
}
```

`process.infd: 4`


0 1 2 3 4
...
0 1 2 3 4
...
term
term
term
READ REFS: 1
WRITE REFS: 1
...