8/9 Questions Answered

Saved at 1:52 PM

Check-in Quiz 01 fork(), signal(), pipe() & file descriptors

Q1
1 Point

If two processes have the same file descriptor number open corresponding to the same file, and one process closes the file descriptor, can the other process still use their file descriptor?

Yes

No

Explanation

Yes, each process has its own copy of the file descriptor table. Closing a file descriptor only closes it in that specific table.

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2
7 Points

For this next problem, assume that when a new file/pipe is opened, that the lowest unused non-negative integer is used for the file descriptor.

Pipe creates two file descriptors, you can assume that pipe_fds index 0 has a lower integer value than pipe_fds index 1

When the program reaches the comment marked "HERE", what does each file descriptor number refer to? (note: HERE is marked by a

comment right before the end of the program, where it is returning from `main()`)

```
int main() {
  int pipe_fds[2];
  pipe(pipe_fds);

  pid_t pid = fork();

  if (pid == 0) {
    close(pipe_fds[1]);
    dup2(pipe_fds[0], STDIN_FILENO);

    char buf[51];

    ssize_t res = read(STDIN_FILENO, buf, 50);

    buf[res] = '\0';

    write(STDOUT_FILENO, buf, strlen(buf));

    exit(EXIT_SUCCESS);
  }

  char buf[50];
  close(STDIN_FILENO);
  int fd = open("hello.txt", O_RDWR);

  dup2(pipe_fds[1], STDOUT_FILENO);

  ssize_t num_read = read(STDIN_FILENO, buf, 50);
  write(STDOUT_FILENO, buf, num_read);

  waitpid(pid, NULL, 0);

  // *************** HERE **************

  return EXIT_SUCCESS;
}
```

Q2.1 fd 0
1 Point

What does the file descriptor `0` refer to when it reaches the `// *************** HERE **************` comment?

Terminal Input

Terminal Output

hello.txt

read end of the pipe

write end of the pipe

unused

Explanation

Correct! after forking, the parent closes STDIN_FILENO, which is `0`, making the file descriptor unused. Then when we open hello.txt, it uses the lowest unused file descriptor, which is now `0`

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2.2 fd 1
1 Point

What does the file descriptor `1` refer to when it reaches the `// *************** HERE **************` comment?

Terminal Input

Terminal Output

hello.txt

read end of the pipe

write end of the pipe

unused

Explanation

Correct! `dup2(pipe_fds[1], STDOUT_FILENO);` makes STDOUT_FILENO, which is equal to `1` to refer to the same thing as `pipe_fds[1]`. `pipe_fds[1]` is the write end of the pipe, so that is what the file descriptor `1` refers to.

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2.3 fd 2
1 Point

What does the file descriptor `2` refer to when it reaches the `// *************** HERE **************` comment?

- Terminal Input
- Terminal Output
- hello.txt
- read end of the pipe
- write end of the pipe
- unused

Save Answer

Q2.4 fd 3
1 Point

What does the file descriptor `3` refer to when it reaches the `// *************** HERE **************` comment?

- Terminal Input
- Terminal Output
- hello.txt
- read end of the pipe
- write end of the pipe
- unused

Explanation

Correct! Initially the first three file descriptors, 0, 1, and 2, refer to STDIN, STDOUT, and STDERR respectively. This means the lowest available file descriptor is 3, so when we call pipe, it uses 3 as the file descriptor for the read end of the pipe.

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2.5 fd 4
1 Point

What does the file descriptor `4` refer to when it reaches the `// ********** HERE **********` comment?

- Terminal Input
- Terminal Output
- hello.txt
- read end of the pipe
- write end of the pipe
- unused

Explanation

Correct! See the answer to fd = 3. After setting up that file descriptor, the lowest available file descriptor available is 4, so pipe() uses 4 as the file descriptor for the write end of the pipe.

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2.6 fd 5
1 Point

What does the file descriptor `5` refer to when it reaches the `// ********** HERE **********` comment?

- Terminal Input
- Terminal Output
- hello.txt
- read end of the pipe
- write end of the pipe
- unused

Explanation

Correct! this file descriptor number is never needed to be used

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q2.7 Output
1 Point

Assuming the file `hello.txt` has the contents:

```
Howdy partner! Might I say you're looking fit as a fiddle.
```

What does the program print, and by which process?

- Parent Process Prints the contents of `hello.txt`
- Child Process Prints the contents of `hello.txt`
- Parent Process Prints the contents of `hello.txt`, and some uninitialized memory
- Child Process Prints the contents of `hello.txt`, and some uninitialized memory
- Nothing is printed or program encounters undefined behaviour

Explanation

Correct! Child prints to stdout_fileno, without having changed where it goes, while reading the file contents from the pipe

Save Answer Last saved on **Oct 09 at 1:52 PM**

Q3 Signal Interruption
2 Points

Consider the following code:

```
bool done = false;

void handler(int signo) {
  printf("h");
  printf("i");
  done = true;
```

```
}


int main() {
  signal(SIGALRM, handler);
  alarm(1);

  printf("V");
  printf("I");

  while(!done) { }

  return EXIT_SUCCESS;
}
```

how many possible different outputs are there? Assume that none of the functions fail.

Please answer with a number only

3

Explanation

Correct! All four characters will always be printed, 'I' will always come after 'V' and 'i' will always come after 'h'. The signal handler could be theoretically invoked anytime after the call to `alarm(1);`, this means we could get any of the three as output: "hiVI", "VhiI", "VIhi". Note that the last output "VIhi" is most likely to happen. The program will most likely be able to print "VI" before a second passes.

Save Answer Last saved on **Oct 09 at 1:52 PM**

Save All Answers **Submit & View Submission ›**