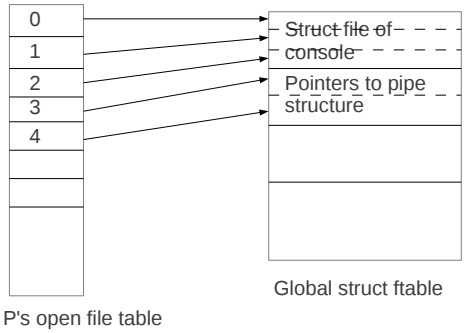
CS347: Operating Systems

Problem Set 4

1. Consider the following snippet of the shell code from xv6 that implements pipes.

```
pcmd = (struct pipecmd*)cmd;
if(pipe(p) < 0)
  panic("pipe");
if(fork1() == 0){
close(1);
dup(p[1]);   //part (a)
close(p[0]);
close(p[1]);
runcmd(pcmd->left);
}
if(fork1() == 0){
close(0);
dup(p[0]);
close(p[0]);
close(p[1]);
runcmd(pcmd->right);
}
close(p[0]);
close(p[1]);  //part (b)
wait();
wait();
```

   Assume the shell gets the command "echo hello | grep hello". Let P denote the parent shell process that implements this command, and let CL and CR denote the child processes created to execute the left and right commands of the above pipe command respectively. Assume P has no other file descriptors open, except the standard input/output/error pointing to the console. Below are shown the various (global and per-process) open file tables right after P executes the `pipe` system call, but before it forks CL and CR.



Draw similar figures showing the states of the various open file tables after the execution of lines marked (a) and (b) above. For each of parts (a) and (b), you must clearly draw both the global and per-process file tables, and illustrate the various pointers clearly. You may assume that all created child processes are still running by the time the line marked part (b) above is executed. You may also assume that the scheduler switches to the child right after fork, so that the line marked part (a) in the child runs before the line marked part (b) in the parent.

2. Consider the execution of the system call `open` in xv6, to create and open a new file that does not already exist.

   (a) Describe all the changes to the disk and memory data structures that happen during the process of creating and opening the file. Write your answer as a bulleted list; each item of the list must describe one data structure, specify whether it is in disk or memory, and briefly describe the change that is made to this data structure by the end of the execution of the open system call.

   (b) Suggest a suitable ordering of the changes above that is most resilient to inconsistencies caused by system crashes. Note that the ordering is not important in xv6 due to the presence of a logging mechanism, but you must suggest an order that makes sense for operating systems without such logging features.

3. Consider the operation of adding a (hard) link to an existing file /D1/F1 from another location /D2/F2 in the xv6 OS. That is, the linking process should ensure that accessing /D2/F2 is equivalent to accessing /D1/F1 on the system. Assume that the contents of all directories and files fit within one data block each. Let $i(x)$ denote the block number of the inode of a file/directory, and let $d(x)$ denote the block number of the (only) data block of a file/directory. Let L denote the starting block number of the log. Block L itself holds the log header, while blocks starting $L + 1$ onwards hold data blocks logged to the disk.

   Assume that the buffer cache is initially empty, except for the inode and data (directory entries) of the root directory. Assume that no other file system calls are happening concurrently. Assume that a transaction is started at the beginning of the link system call, and commits right after the end of it. Make any other reasonable asusmptions you need to, and list them down.

   Now, list and explain all the read/write operations that the disk (not the buffer cache) sees during the execution of this link operation. Write your answer as a bulleted list. Each bullet must specify

whether the operation is a read or write, the block number of the disk request, and a brief explanation on why this request to disk happens. Your answer must span the entire time period from the start of the system call to the end of the log commit process.

4. Provide one reason why a DMA-enabled device driver usually gives better performance over a non-DMA interrupt-driven device driver.

5. Which of the following statements is/are true regarding memory-mapped I/O?

Answer _____

A. The CPU accesses the device memory much like it accesses main memory.

B. The CPU uses separate architecture-specific instructions to access memory in the device.

C. Memory-mapped I/O cannot be used with a polling-based device driver.

D. Memory-mapped I/O can be used only with an interrupt-driven device driver.

6. Which of the following statements is/are true regarding the file descriptor (FD) layer in xv6?

Answer _____

A. The FD returned by `open` is an index to the global `struct ftable`.

B. The FD returned by `open` is an index to the open file table that is part of the `struct proc` of the process invoking `open`.

C. Each entry in the global `struct ftable` can point to either an in-memory inode object or a pipe object, but never to both.

D. The reference count stored in an entry of the `struct ftable` indicates the number of links to the file in the directory tree.