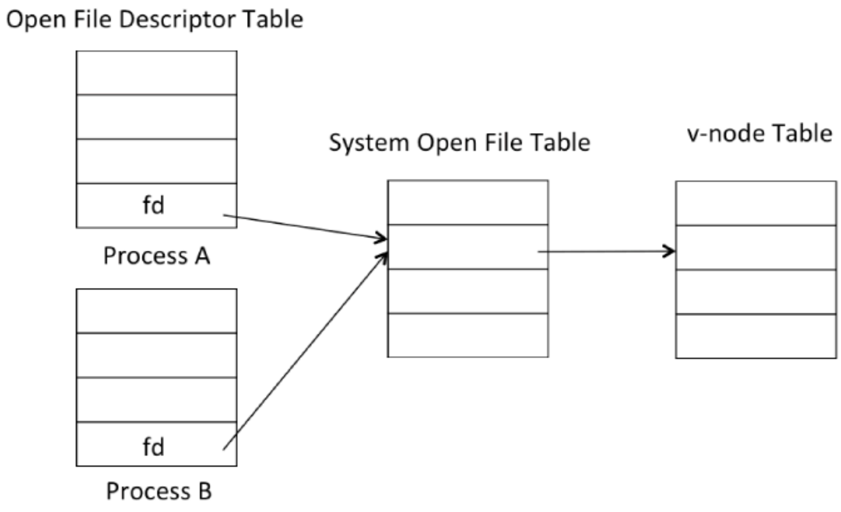
System Programming (Fall, 2016)
Hand-written Assignment 1 (Due on 10/12, in class)

1. **Atomic operation**. In class we explained why file appending, i.e., the positioning to the current end of file and the write, should be an atomic operation. Suppose there are two processes A and B trying to append to the same file. Each has opened the file but without the O_APPEND flag. The following figure shows the relationship among the three tables, in which each process refers to the file by its own file descriptor *fd*. Different from the case we discussed in class, the two file descriptors point to the same system open file table entry.

   (a) In the new scenario, is it still necessary to have the positioning to the current end of file and the write be an atomic operation? Give an example to explain your answer.



   (b) Another way to write to a file descriptor at a given offset is to call pwrite(), whose prototype is defined as follows:

```
ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset).
```

   pwrite() writes up to *count* bytes from the buffer starting at *buf* to file descriptor *fd* at offset *offset*. The file offset for *fd* will not be changed by this system call. pwrite() might be composed of multiple steps and coded as follows:

```
ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset)
{
    off_t currpos;
    int retval;
    currpos = lseek( fd, 0, SEEK_CUR);
    if ( lseek( fd, offset, SEEK_SET) < 0 ) return -1;
    retval = write( fd, buf, count );
    lseek( fd, currpos, SEEK_SET);
    return retval;
}
```

   Based on the scenario given in (a), please give an example to explain why pwrite() should be an atomic operation.