Simon Yoon & Joya Debi
ECE357: Computer Operating Systems
Professor Hakner
WS01

1.

a) Reading 10 bytes from a file on disk

   Always, getting file permissions to create, delete, open, close, read, write, etc. is a system call and in order to read, a system call would trigger.

b) Allocating memory via malloc

   Sometimes. malloc is in lib.c and has a pre-allocated memory threshold, MMAP_THRESHHOLD, defaulted to 128 kB. In the case that the memory being allocated is less than the threshold of 128 kB, no system call occurs. However, if the demand requires more memory than MMAP_THRESHHOLD, a system call will be triggered.

c) Calling sqrt(100.0)

   Never, sqrt is in lib.c and there is no condition that would require a system call

d) Getting the current time of day

   Always, the time of day pertains to information maintenance and getting or setting system information which informs the need for communication with the kernel, a consequence of the system call.

2.

a) char buf[10];n=read(-1,buf,100);

   Initializes a char array size 10 filled with random contents in each element since the contents of the array were not initialized. The read function typically has 3 main input parameters. The first input parameter is the file descriptor which cannot be a negative value. It would, in theory, attempt to read 100 bytes from that file descriptor, however with the size of the array being less than 100, it will not compile.

b) fd=open("/ooopsy",O_WRONLY|O_CREAT|O_TRUNC,0666);

   The file descriptor, fd, is being initialized. The open system call opens the file specified by the pathname “/ooopsy”. The argument flags are set to the access mode of “O_WRONLY” which indicates this file will be “write-only”. The file creation flag is set to “O_CREAT” which means if the pathname does not already exist, a regular file will be created with the “/ooopsy” pathname. “O_TRUNC” describes when the file already exists

in said pathname, is regular, and is in either “O_RDWR” or “O_WRONLY”, it will be truncated to length 0. It is ignored if the file in question is FIFO or a terminal device file. 0666 gives read/write permissions to the file, but not execute.

c) char buf[3]; for(;;) printf("%d",read(fd,buf,3)); /* fd is open for reading, the associated file is 9 bytes long */

char buf[3] initializes a char array size 3 filled with random contents in each element. The for(;;) statement has no initial or test condition, so the loop will run forever. The read system call will return the number of bytes read in the file indicated by fd and the printf function will then print those values. The first three iterations will return 3. Then, the file descriptor points to the end of the file as all 9 bytes of the file have been read. Afterwards, the read system call will return 0 for all future iterations. What will be printed is “3 3 3 0 0 0 0 0 …”

d) fd=open("/tmp/good",O_WRONLY|O_APPEND|O_TRUNC|O_CREAT,0666); /*assume it succeeds*/
write(fd,"hewitt\n",6);
lseek(fd,0,SEEK_SET);
write(fd,"8675309\n",8);
p=lseek(fd,0,SEEK_END);
printf("%d\n",p);

The file descriptor, fd, is being initialized here as well. The open system call opens the file specified by the pathname “/tmp/good”. The argument flags are set to access mode of “O_WRONLY” which indicates this file will be “write-only”. Assuming “O_APPEND” doesn’t corrupt files on the NFS file systems, the file is opened in append mode. ”O_TRUNC” truncates the file to length 0 if the file already exists, is regular, and is in either “O_RDWR” or “O_WRONLY”. Otherwise, it is negligible. O_CREAT again creates a regular file in the pathname “/tmp/good” in the absence of an identical file in existence prior. 0666 gives read/write, but not execute permissions. Assuming the file descriptor initialization succeeds, write(fd, “hewitt\n”,6); writes up 6 bytes in the form of “hewitt” onto fd, \n counts as 1 byte and is excluded. lseek repositions fd to the offset number of bytes as indicated by SEEK_SET at 0. write(fd, “8675309\n”,8); writes up 8 bytes in the form of “8675309” starting at 0 and the newline is included, thus pushing “hewitt” to the next line. p is being initialized. The file offset is set to the length of the file with “SEEK_END”, adding 0 to the size of the file as the file offset. In printf(“%d\n”,p);, printf prints an int on newline, taking in argument p. p was previously initialized to be the size of the file which is 14 bytes.

The file contains:	The terminal prints:
8675309 hewitt	14