FIT2100 Tutorial #3 I/O Management, and Disk Scheduling (Suggested Solutions) Week 11 Semester 2 2019

August 21, 2019

Revision Status:

`$Id: FIT2100-Tutorial-06.tex, Version 2.0 2018/10/02 13:45 Jojo $`

`Adapted to new tutorial by Daniel Kos`

Contents

1 Pre-class exercises

(a) The problem with the program is that the returned array was only created as a local variable within the function that returned it. Once the function returns, the array is deallocated from the stack, and the returned pointer is useless.

(b) Change `int series[SERIES_SIZE];` to `static int series[SERIES_SIZE];`. Now the array is allocated on the heap and will not be destroyed when the function returns.

2 I/O Management and Disk Scheduling

2.1 Review Questions

Question 1

- **Programmed I/O:** The processor issues an I/O command, on behalf of a process, to an I/O module; the process then *busy-waits* for the operation to be completed before proceeding.
- **Interrupt-driven I/O:** The processor issues an I/O command on behalf of a process, continues to execute subsequent instructions, and is interrupted by the I/O module when the latter has completed its work. The subsequent instructions may be in the same process, if it is not necessary for the process to wait for the completion of the I/O.

Otherwise, the process is suspended pending the interrupt and other work (or another process) is performed.

- **Direct memory access (DMA):** A DMA module controls the exchange of data between main memory and an I/O module without the intervention of the processor. The processor sends a request for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

Question 2

- **Block-oriented devices:** store information in blocks that are usually of fixed size, and transfers are made one block at a time. Generally, it is possible to reference data by its block number. Disks and tapes are examples of block-oriented devices.
- **Stream-oriented devices:** transfer data in and out as a stream of bytes, with no block structure. Terminals, printers, communications ports, mouse and other pointing devices, and most other devices that are not secondary storage are stream oriented.

Question 3

Double buffering allows two operations to proceed in parallel rather than in sequence. A process can transfer data to (or from) one buffer while the operating system empties (or fills) the other.

Question 4

The delay elements which are involved in a disk read or write include: seek time, rotational delay, and access time.

Question 5

- **FIFO:** Items are processed from the queue in sequential first-come-first-served order.
- **SSTF:** Select the disk I/O request that requires the least movement of the disk arm (head) from its current position.
- **SCAN:** The disk arm moves in one direction only, satisfying all outstanding requests en route, until it reaches the last track in that direction or until there are no more requests in that direction. The service direction is then reversed and the scan proceeds in the opposite direction, again picking up all requests in order.
- **C-SCAN:** Similar to SCAN, but restricts scanning to one direction only. Thus, when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again.

Question 6

Internal fragmentation means that a file does not fill an entire block. Since the disk is addressed by block, the space left over can't be filled by a different file, so the space is wasted.

External fragmentation affected older filesystems where blocks in a file had to be allocated contiguously. If there weren't enough free blocks in a row to fit a file, that space could not be used. Solutions include chaining (each block contains a pointer to the next block where the file continues) and inodes (an index node stores a set of pointers to different blocks in the file). These approaches mean that blocks are never wasted, however if the blocks in the file are not nearby, performance can suffer due to needing to seek to different tracks on the disks.

2.2 Problem-Solving Tasks

2.2.1 Task 1

(a) (i) **FIFO**: `27, 129, 110, 186, 147, 41, 10, 64, 120`
(ii) **SSTF**: `110, 120, 129, 147, 186, 64, 41, 27, 10`
(iii) **SCAN**: `64, 41, 27, 10, 110, 120, 129, 147, 186`
(iv) **C-SCAN**: `64, 41, 27, 10, 186, 147, 129, 120, 110`

(b) Average seek length (in terms of the number of tracks traversed):

(i) **FIFO**: `61.8`
(ii) **SSTF**: `29.1`
(iii) **SCAN**: `29.6`
(iv) **C-SCAN**: `38`

2.2.2 Task 2

Each sector (512 bytes) can hold 4 logical records (128 bytes per record). The required number of sectors is `300,000/4 = 75,000` sectors. This requires `75,000/96 = 782` tracks, which in turn requires `782/110 = 8` surfaces.