PDS/SDP OS internals 7/7/2023 – STANDARD version (with proposed solution)

PDS: L'esame è comune ai corsi PDS e SDP (in inglese). Si è predisposta un'unica proposta di soluzione in inglese.
SDP ON/OFF: As the ON/OFF exam is a subset of the standard one, the ON/OFF solution can be easily extracted from the standard one.

1 Consider the following string of memory references, for a given process. For each reference (Byte addressing, with addresses expressed in hexadecimal code) the read(R)/write(W) operation is also reported: R 33FB, R 1B64, W 30D3, W 237E, R 0AC8, W 23D7, R 174A, R 0965, W 32A0, R 1BB0, W 09E5, R 3380, R 2A94, R 11B8. Assume that physical and logical addresses are on 16 bits, page size is 2KBytes, and 70FF is the maximum address usable by the program (the address space top limit).

A) Compute the size of the address space (expressed as number of pages), and the internal fragmentation.

The address space goes from 0 to 70FF, so its size is 7100 = 28KB + 256B Npages (28KB + 256B)/2KB = 15 pages. Int frag = 2KB-256B = 1792B

B) Compute the string of page references.

6	3	6	4	1	4	2	1	6	3	1	6	5	2

C) Simulate a **second-chance** page replacement algorithm, with 3 available frames. Represent the resident set (physical frames containing logical pages) after each memory reference. Explicitly indicate, for each allocated frame, the page number and the reference bit (use the p,r or the p_r notation). Also indicate page faults Assume that the reference bit of a page is initialized to 0 after a page fault.

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13
References	6	3	6	4	1	4	2	1	6	3	1	6	5	2
Resident Set	6_0	6_0	6_1	6_1	6_0	6_0	2_0	2_0	2_0	3_0	3_0	3_0	5_0	5_0
		3_0	3_0	3_0	1_0	1_0	1_0	1_1	1_0	1_0	1_1	1_1	1_0	2_0
				4_0	4_0	4_1	4_0	4_0	6_0	6_0	6_0	6_1	6_0	6_0
Page Faults	*	*		*	*		*		*	*			*	*

2 Consider a Hard Disk Drive (HDD) with standard technology and geometry. The disk has

- sector size of 512 Bytes,
- 2048 tracks per surface,
- 50 sectors per track,

- 5 double-sided platters,
- average seek time of 10 msec.

A) What is the capacity of a track (in Bytes)? What is the capacity of each surface? What is the capacity of the disk? How many cylinders does the disk have?

Bytes/track = Bytes/sector * sectors/track = 512B * 50 = 25KB
Bytes/surface = Bytes/track * tracks/surface = 25K * 2048 = 50MB
Bytes/disk = Bytes/surface * surfaces/disk = 50MB * 5 * 2 = 500MB
The number of cylinders is the same as the number of tracks on each platter, which is 2048.

B) The CHR (Cylinder-Head-Sector) disk addressing method allows a disk block to span over multiple sectors of a given track (identified by a cylinder,head pair), but not on different tracks. Are the following valid block sizes: 256 B, 2048 B, 51200 B? (motivate the answers)

The block size should be a multiple of the sector size and less-equal than the track size..
- 256B_ NO. not a multiple of the sector size
- 2048B: YES, It is a multiple of a sector
- 51200B: NO, exceeds the size of a track, which is 25KB

C) If the disk platters rotate at 5400 rpm (revolutions per minute), what is the maximum rotational delay? What is the average rotational delay? If one track of data can be transferred per revolution, what is the transfer rate (expressed in bits/second)? What would be the transfer rate if one entire cylinder of data could be transferred per revolution?

If the disk platters rotate at 5400rpm, the time required for one complete rotation, which is the maximum rotational delay, is:
(1/5400) * 60 s = 11ms.
The average rotational delay is half of the rotation time.5,5ms
The capacity of a track is 25K bytes. Since one track of data can be transferred per revolution, the data transfer rate is:
25K/((1/5400) * 60) B/s = 90*25KB/s = 2.25MB/s = 2.25*8 Mbit/s = 18 Mbiit/s
One cylinder per revolution: transfer rate is multiplied by tracks/cylinder (10) -> 18*10 Mbit/s = 180 Mbit/s

3 An application needs to organize a file with a header and multiple sections. The header includes, among other data, offsets, and sizes of all sections in the file.

A) for each of the questions below, answer yes/no and motivate the answer

		Yes/No	Motivate/explain
1	Can the application assume that the header, as well as each section, be aligned with (stored starting at) disk block boundaries in the file system?	NO	Because file allocation in blocks is handled at a different software layer (file organization module), which as little to do with the application layer, unless (this is more of an exception) the file format is a standard, willingly supported and handled by the file organization layer

2	Can the application organize data in multiple files (instead of one single file): one file for the header and one for each section? (If NO, motivate, if YES, motivate and explain how section offsets in the header should be represented, encoded, or changed)	YES	Or course it can. The solution is obviously different from the single file one. Offsets are not meaningful anymore, as sections are in different files. So, references to sections could be, for instance, file names.

B) A text file is copied to three different file systems, whose file formats are, respectively:

- contiguous allocation (by multiple of a disk block)
- linked allocation (no FAT)
- Inode

Answer the following question and motivate the answer:

		Yes/No	Motivate/explain
1	The size of a disk block is the same on all three file systems. Can we assume that the internal fragmentation will be the same on all file systems?	NO	Because with linked allocation a data block can hold fewer Bytes that with contiguous and Inode.

C) A file system contains three text files a.txt, b.txt and c.txt. A file archival/compression application (such as for instance gzip, tar, 7z, rar, etc.) stores the (content of) the three text files within a single archive file (e.g. abc.zip).

For each of the questions below, answer yes/no and motivate the answer

		Yes/No	Motivate/explain
1	Can the three text files share the same disk block, which means that the file system is storing them (at least partially) within the same disk block?	NO	Because they are three different files, so the file system is internally allocating each of them in a dedicated set of disk blocks.
2	Is it possible, for the content of the three text files, replicated and/or compressed in the archive file, to share (at least partially) the same disk block (so to be stored within the same disk block)?	YES	Because it is a single file, so, while not guaranteed, it is possible that Bytes coming from different original files converge into the same disk block.

4 An OS161 system is given. Consider the file system framework used in LAB5, partially represented by the following code excerpts:

Code excerpts from files proc.h and file_syscalls.c

```
struct proc {
        ...
#if OPT_FILE
        /* per-process open file table */
        struct openfile *fileTable[OPEN_MAX];
#endif
};
```

```
/* system open file table */
struct openfile {
```

```
  struct vnode *vn;
  off_t offset;
  unsigned int countRef;
};

struct openfile systemFileTable[SYSTEM_OPEN_MAX];
```

Answer the following questions:

A) Why is systemFileTable an array of struct openfile, whereas fileTable is an array of pointers to struct openfile?

A struct openfile is a data structure associated created when opening a file: each openfile struct holda one offset for reading into a file. Two processs sharing the offset when reading/writing will share the openfile struct, whereas if they use independent offsets, they will use different openfile struct. In order to allow sharing (or not sharing) of openfile structs, they are allocated into the system wide table.

Processes can only point to open file structs, so that they can either share them or not.

B) Is the countRef field redundant, considering that the vnode pointed by the vn field has its own reference count?

No, it is not redundant. The vnode reference count counts how many vfs_open have been done (so how many offsets are active on the file), whereas countRef in openfle struct count how many processes are sharing the offset (and the openfile struct holding it.

C) In case we need to support file locking, what should we choose between a spinlock and a lock, and where should it (or they, if multiple), be located: in systemFIleTable, in fileTable, or elsewhere?

Unless we go for a low leven re-mplementation of a lock, lock should be preferred to spinlocks, because file locking would not be adequate for busy waiting (file operations are not fast enough).

The lock needs to be accessible/used by all processes working on a given file. As a lok is a dynamically allocated data structure, it is enough to create it once: pointers to the lock caoud be either in the openfile struct or in the prt-process table, depending on the locking sckeme adopted.

D) What is the possible role of functions copyin and copyout when implementing the sys_read and sys_write system calls?

copyout() and copyin()are used to copy data between user memory and kernel memory safely, that is, without the kernel crashing in case of an incorrect user pointer. They can be used while implementing sys_read andsys_write, provided that I7O is done through kernel buffers.

5 Consider, in OS161, the two possible implementations of spinlock_acquire shown below:

```
void spinlock_acquire(struct spinlock *splk) {
  // ...
  while (1) {
```

```
    if (spinlock_data_get(&splk->splk_lock) != 0) {
      continue;
    }
    if (spinlock_data_testandset(&splk->splk_lock) != 0) {
      continue;
    }
    break;
  }
  // ...
}
```

```
void spinlock_acquire2(struct spinlock *splk) {
  // ...
  while (1) {
    while (spinlock_data_get(&splk->splk_lock) != 0);
    if (spinlock_data_testandset(&splk->splk_lock) == 0) {
      continue;
    }
  }
  // ...
}
```

A) Why is spinlock_acquire using both spinlock_data_get and spinlock_data_testandset, instead of calling just one of them?

Because the true busy waiting cycle is going to be done on spinlock_data_get, which is less expensive, in term of impact on system performance, of spinlock_data_testandset. A loop on spinlock_data_get alone cannot be done as it does not provide atomic test-and-set.

B) Is spinlock_acquire2 equivalent to spinlock_acquire? If Yes, motivate, if no, motivate and, if possible, modify is in order to make it equivalent

NO. As far as loping on spinlock_data_get is considered, they are equivalent, but when coming to spinlock_data_testandset, spinlock_acquire exits the loop (correctly) when the function returns 0, whereas spinlock_acquire2 loops infinitely. The continue; statement has to be replaced by a break; statement. Then the two functions are equivalent.