VauLSMorg
09 FSStore - File System & Persistent Storage
Aneka Soal Ujian Sistem Operasi
Rahmat M. Samik-Ibrahim et.al.

1. **2016-1a**

   Circle or cross: "**T**" if True – "**F**" if False.

   T / F A file is logical storage unit (Silber9).

   T / F A volume (of file system) may be a subset of a device, or a whole device, or multiple devices linked together into a disk array set (Silber9).

   T / F Microsoft Windows' volume label "C:" is usually reserved for the main disk. Label "A:" and "B:" were once reserved for the floppy disks.

   T / F The implementation of File Systems on Virtual Machines is called Virtual File Systems (VFS (Silber9).

   T / F One disadvantage of linked allocation method (of disk space) is external fragmentation (Silber9).

   T / F A unified buffer cache can not solve the problem of double caching (Silber9).

2. **2016-1b**

   Diketahui sebuah disk dengan 100 silinder (0 - 99) menggunakan algoritma penjadwalan C-LOOK dengan antrian (queue) terpisah untuk "menulis" (W) dan "membaca" (R) sebagai berikut:

   - antrian "R":
     selama tidak kosong, hanya antrian ini yang akan dilayani (kecuali jumlah W tertentu).
   - antrian "W":
     hanya dilayani, jika antrian R kosong. Kecuali:
   - antrian "W" menumpuk lebih dari 10:
     maka antrian R harus menunggu hingga satu siklus C-LOOK penuh.
   - UP:
     Untuk pergerakan antar silinder (UP), diperlukan 1 unit waktu.
   - RETURN, RtoW, WtoR:
     Untuk balik (return), switch dari R ke W atau dari W ke R, diperlukan 5 unit waktu.
   - Sekali heads bergerak, permintaan baru tidak akan mengubah tujuan dari heads (hingga heads sampai tujuan).
   - Saat T=0, posisi heads pada silinder 0.
   - Abaikan "rotational latency".

   Permintaan akses sebagai berikut:

Time(t)	000	020	040	060	080	100	120	140	160	180	200
R	50	20	40	60	80	-	50	-	-	-	-
W	-	-	-	-	20	-	-	-	-	-	-

   Lengkapi table berikut (tersedia 2 baris contoh pengisian):

Silinder	Time	R-QUEUE	W-QUEUE	QUEUE	NEXT
00	000	50	-	R-QUEUE	UP
50	050	20:40	-	R-QUEUE	RETURN
Waktu Total					

3. **2016-2**

Consider 24 disks (@ 1TB) in a RAID 6+1 formation: D01, D02, D03, D04,.... D23, D24.

(a) Draw the RAID 6+1 diagram! Do not forget to give proper labels to show the RAID6/RAID1 parts.

(b) What is that storage capacity (TB) of the RAID 6+1 above?

(c) What is the Read speed up?

(d) What is the Write speed up?

4. **2017-1**

Circle or cross: "T" if True – "F" if False.

T / F There is no external fragmentation in a file system with linked allocation.

T / F The Deadline I/O Scheduler (Linux) gives the **Read Queues** a higher priority.

T / F In a distributed file system, it is possible to write unnoticed by others for a short time.

T / F Doubling the block size in a indexed allocation disk space system will exactly double the maximum file size.

DISCLAIMER: These following are logical (not physical) numbers! Consider a disk with 10000 cylinders (cyl. 0 to 9999) and 2 surfaces. Each track has 100000 sectors. Each sector size is 1000 bytes. The spin rate is 6000 RPM (Revolutions Per Minute). A seek takes 1 milliseconds per cylinder moved. The initial disk head position is at cylinder 0. Assume that 1 GBytes = 1000 Mbytes = 1000000 KBytes = 1000000000 Bytes.

T / F The spin rate 6000 RPM is known in SI (Systeme Internationale) as 100 Hz.

T / F Each full disk rotation will take 10 ms.

T / F There will be 100 Mbytes data in each track.

T / F The maximum theoretical transfer rate will be 100 Mbytes/10 ms = 10 GBytes/ second.

T / F Each surfaces of that disk will have 5000 tracks.

T / F The total disk capacity will be 10000 GBytes.

5. **2017-2**

(Adapted from JJ Pfeiffer, "Writing a FUSE Filesystem: a Tutorial", NMSU, licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License)

One of the real contributions of Unix (and later Linux) has been the view that "everything is a (01)". A tremendous number of radically different sorts of objects, have been mapped to the (02). One of the more recent directions this view has taken, has been (03) or also known as (04). A (05) is a program that listens on a (06) for file operations to perform, and performs them. With FUSE, (07) users can create their own file systems. The idea here is that users can write a FUSE file system to provide interaction with an object in terms of a (08) and (09). A user just has to write codes that implements file operations like (10), (11), and (12).

Match the number of the sentence above with these following phrases:

[ ] directory structure	[ ] file	[ ] file abstraction	[ ] filesystem operations
[ ] Filesystems in User Space	[ ] FUSE	[ ] FUSE filesystem	[ ] non-privileged
[ ] open()	[ ] read()	[ ] socket	[ ] write()