2020 Operating System Examination Solution

1. a. Monolithic kernel puts almost all core operating system functionalities into one single executable with its own address space, whereas microkernels keep only a minimal set of operating system responsibility in the kernel and everything else, such as file system, device drivers, into the user space. Monolithic kernel makes easier for different kernel components to communicate with each other and allows faster system calls than microkernels due to shared memory space, but it can become over-complicated and vulnerable to component crashes. (e.g. once a single component crashes, it might bring down the entire kernel) On the other hand, microkernels are less complicated and more structured. Microkernels will not crash if one of the component does not work properly. However, the overhead of inter-process communication within kernel is high, thus it might be slower than a typical monolithic kernel.

   b. When the bug occurs in a monolithic kernel, it might bring down the entire kernel due to the incorrect memory access, or the monolithic kernel might still be running but its memory gets corrupted and the kernel becomes insecure. Then the bug happens in a microkernel, it will just terminate the proceses related to the file system driver. This will not affect other processes and will not bring down the entire kernel.

   c. The Linux kernel will assign the `/mnt` as the mount point of the experimental file system, with the content of `/mnt` becoming the content of the root directory of the experimental file system. A new entry will be added to the mount table to indicate which device and what type of file system is mounted. Upon visiting `/mnt`, the experimental file system driver will be called and the mount table will be used in order to visit this new file system.

   d. i) Level 0. Since the storage is temporary and the faster the better for big data computation, it is better to use RAID storage with level 0

   ii) Level 5. Since we need the content in the web server to be highly-available, we need to make sure that there is a reasonable amount of redundancy, yet not too costly. We do not choose level 1 here because it costs more.

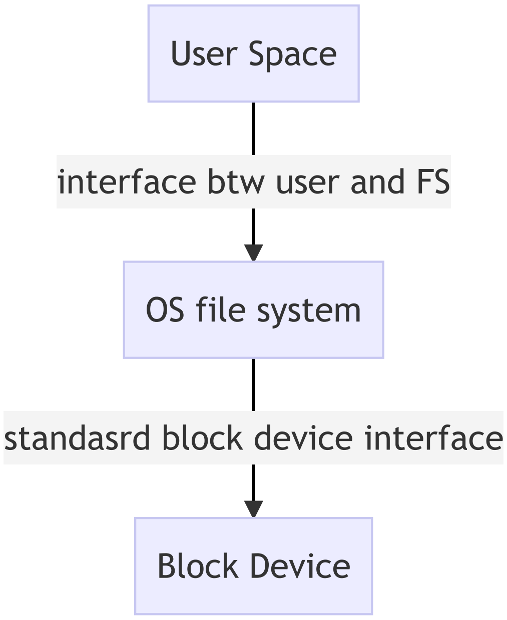
   iii) Level 1. It is crucial for a database to retain its data while keeping the reading/writing speed faster. Though more costly, level 1 offers a better recovery time(in fact no recovery is needed since the other disk contains the same copy of data) and a better write time than level 5.

   e. Below is a possible interface for implementing an experimental file system

```
// below is a list of functions that belows to the interface between user
space and file system
create()
link()
unlink()
open()
read()
write()
close()
readdir()
...
```

When the user creates a file using the interface above, the OS will call the corresponding function in the file system driver, so that the file system driver can handle the operation.

```
// below is a list of functions from the Linux standard block device struct
read()
write()
seek()
...
```



As shown above, the kernel can support different file system and RAID implementations by first overriding the standard block device interface, then overriding the interface between user space program and file system. There are several key data structures in this process as listed below

```
// block device struct
struct block_device {
  int device_type;
  char* device_name;
  ...
  file_operations** operations; // all the operation functions of the
standard block device
}

// the parameters of the functions below have been omitted for clarity
struct file_operations {
  (void *) read();
  (void *) write();
  (void *) create();
}

// this struct is used between user and file system
struct fs_operations {
  (void *) create();
  (void *) link();
  (void *) unlink();
  (void *) open();
  (void *) read();
  (void *) write();
  (void *) close();
  ...
}
```

2. a. i) Alice and Bob can write to `File2` while Dave, Edward and Fred can read and execute `File2`

   ii) Carol can ask either George or Henry, both of who are in D4, to give her the `write` access right to GPU device

   iii) The attacker can now read or write `File1`, write `File2`, read/write/execute `File3` and write to GPU device. The attacker can become either of D4, D2, or D1 user since D4 users can *enter* D2 and D2 users can *enter* D1.

   iv) When Henry is no longer a member of the domain D4 *OR* a user from D3, who is the owner of the GPU device, has changed the access right of D4 from `write*` to none.

   v) In the column for `File2` and row for D1, we change the access right from `write` to `write*` and then create another domain D5 that contains all the access right of George inherited from D4, then remove George from D4 and add him to D5. Now, Alice can delegate `write` to `File2` only to Henry

   b. One way is to implement it using Access Control List. Each file/object will contain a list of principal that can read/write/execute it. Access Control List has the advantage of avoiding duplication when there is a large amount of users with the same priviledges since using capability will result in lots of users having the same capability list. It is also easier to manage user priviledges as deleting/adding priviledge to an object is easier.

Another way is to use capabilities. Each user/principle maintains its own list of actions/objects to read/write/execute. Capability makes it easier the principle of least privilege as we can define a user's capability by adding the minimal amount of files the user actually needs. It is also easier to trsansfer the rights of one user to another. However, it is harder to revoke the rights of the principles because it is hard to ensure that all the capability tokens/bit string have been destroyed.

c. We need to ensure that the principle of least privilege has been maintained so that the attacker will not gain access to files at a higher level. If necessary, we can use Mandantary Access Control so that every object/file's access is mandated by the operating system rather than other users, as opposed to discretionary access control. In this way, we can prevent other persuaded employee to alter the access right of objects and allow the attacker to visit them.