SUB 2018- 2019

1. Write a unix shell command that displays the lines in a file that contains words starting with capital letters.

```
grep -h '\<[A-Z][a-zA-Z0-9]*\>' a.txt
```

2. Write a unix shell command that inverts in a file b.txt all pairs of neighboring digits

```
sed 's/\([[:digit:]]\)\([[:digit:]]\)/\2\1/g' b.txt
```

3. File c.txt contains on each line two numbers separated by space. Write a unix shell command that displays for each line the sum of its numbers

```
awk '{s=0; for(i=1;i<=NF;i++){s+=$i}print s}' c.txt
```

4. Display only the lines of a file d.txt that appear only once.

```
sort d.txt | uniq -u
```

5. Write a unix shell script that displays the name of each .txt file in the current directory, that contains the word "cat"

```
#!/bin/bash

for item in $(ls "."); do
        f="./$item"
        grep -l "cat" $f
done
```

6. In the program fragment bellow, mark which process executes each line: the Parent, the Child, or both.

```
P       k = fork();
P C     if (k == 0){
C               printf("A\n");
        }
P       else {
P               printf("B\n");
        }
P C     printf(C\n");
```

7. How many processes will be created by the code fragment below, excluding the initial parent process?

```
fork(); wait(0); fork(); wait(0); fork();
```

7

8. What are the possible console outputs of the following code fragment (ignoring any output that execl might generate), and when will they happen?

printf("A\n"); execl(....); printf("B\n");

A\nB\n when the execl is not executed or only A\n in all other cases

9. What does the system call "read" do when the pipe is empty?

- it returns EOF(return value 0) if no process have the right end open, or if some other process has the pipe open for writing, read will block the anticipation of new data, so this code output hangs because here write ends parent process and also child process don't close

10. What does the system call "open" do before returning from opening fifo?

Waits for the fifo to be open for complementary operation before returning

11. Give a reason for choosing threads over processes

- much quicker to create a thread than a process

- much quicker to switch between threads than to switch between processes

- Inter-thread communication can be faster than inter-process communication because threads of the same process share memory with the process they belong to

DISADVANTAGES

- one thread can stomp on another thread's data
- if one thread blocks, all threads in task block

12. Consider that functions "fa" and "fb" are run in concurrent threads, what will the value of "n" be after the threads are finished? Why?

```
pthread_mutex_t a, b;
int n = 0;

void* fa(void* p){
	pthread_mutex_lock(&a);
	n++;
	pthread_mutex_unlock(&a);
}

void* fb(void* p){
	pthread_mutex_lock(&b);
	n++;
	pthread_mutex_unlock(&b);
}
```

RASP: no of threads

13. Schedule the following jobs (given as Name/Duration/Deadline) so that they all meet their deadlines: A/5/9, B/7/13, C/1/10

A, C, B no delay

14. Give one advantage and one disadvantage of the segmented allocation method over paged allocation method

ADVANTAGES:

- Segment tables use less memory than paging
- The segment table is of less size compared with the page table in paging
- Does not offer internal fragmentation, while paged allocation does
- Simple to relocate segments than the entire address space

DISADVANTAGES:

- Un-equal size of segments is not good in the case of swapping.
- It demands programmer intervention.
- It is hard to allocate contagious memory to partition as it is of its variable size.
- This is costly memory management algorithm.

15. When would you load into memory the pages of a program that is being stared?

I would load a page into memory when it is requested, because if I load them all at once the pagging effect would disappear

16. When does a process change state from RUN to READY?

Once the currently running process's time is up or task is complete or other condition is met, the current process is placed into ready queue (if there are some more instructions pending for that process i.e., if it is done only for time being) else completed (if it is 100% complete). In either of these cases, the first process in the ready queue is popped out from the queue and executed.

17. Given a unix file system configured with a block size of B bytes that can contain A addresses, and i-nodes having S direct link, one simple indirection link, and one triple indirection link, give the formula for the maximum file size possible.

N^2+n^3

18. What happens with the data when you delete a file that has a hard link pointing to it?

When you create hard link to a text file and then you delete the text file, it doesn’t happen anything with the hard link, it keeps the data from the deleted file.

If you delete the hard link, the original file is deleted

19. Give a method from preventing deadlocks.

Deadlocks can be prevented by preventing at least one of the four required conditions:

Eliminate circular wait: choose an order for your resources and always lock them in the same order

Eliminate Mutual Exclusion

- Shared resources such as read-only files do not lead to deadlocks, but it's not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable

Eliminate Hold and wait

- Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization.
- The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

Eliminate Circular Wait

- One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.

Eliminate No Preemption

- Preemption of process resource allocations can prevent this condition of deadlocks, when it is possible.
    - One approach is that if a process is forced to wait when requesting a new resource, then all other resources previously held by this process are implicitly released, (preempted), forcing this process to re-acquire the old resources along with the new resources in a single request, similar to the previous discussion.
    - Another approach is that when a resource is requested and not available, then the system looks to see what other processes currently have those resources and are themselves blocked waiting for some other resource. If such a process is found, then some of their resources may get preempted and added to the list of resources for which the process is waiting.

20. What is a binary semaphore and what is the effect of its P method, when called by multiple concurrent processes/threads?

A binary semaphore is also known as mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.

Method P: If the semaphore is unlocked, it becomes locked and the call returns, allowing the thread to continue. If the semaphore is locked, the thread is blocked and placed in a queue *Q* of waiting threads.

SUB27IUN2019

2. Write a unix shell command that inverts in a file b.txt all pairs of odd digit followed by vowel

sed 's/\([13579]\)\([aeiou]\)/\2\1/g' b.txt

RESTANTA IUL2019

6. How many processes will be created by the code fragment below, excluding the initial parent process?

```
If (fork() || fork()) {
        fork();
}
```

RASP: 4

7. How many processes are created by the code fragment below, when the parent process P calls f(3)?

```
void f(int n){
        if (n > 0 || fork() == 0) {
                f(n-1);
                exit(0);
        }
        wait(0);
}
```

It doesn’t stop

9. What does the system call “read” do when the pipe contains less data than it is required to read, but it is not empty?

It reads the existing data

10. What will the fragment code below print to the console?

```
int p[2];
char buf[10];
int n;
pipe(p);
n = read(p[0], buf, 10);
printf(“%d\n”, n);
```

11. Why are zombie processes problematic?

Zombie processes are problematic because if they are not "waited" they will grow in number, occupying PIDs and bringing the system to a point where no other processes can be created because of lack of PIDs.

14. Give an advantage and a disadvantage of the set-associative caches versus the direct caches.

AD: Set associative caches generally have lower miss rates than direct caches of the same capacity, because they have fewer conflicts / a bit faster

DIS: set associative caches are usually slower and somewhat more expensive to build because of the output multiplexer and additional comparators. / may cause cache collisions, which being frequent is known as cache trashing

15. What page has the highest priority in the LRU replacement policy, when choosing a victim page?

LRU has an N x N matrix of bits (N-number of pages) which is filled with 1 and 0, and the highest priority when choosing a victim page are those pages which have the minimum line sum.

16. Given two set-associative caches, one with 2 sets of 4 pages and one with 4 sets of 2 pages, which would perform better for the following sequence of page request: 20, 9, 18, 27, 20, 9, 18, 27. Why?

4 sets of 2 pages perform better because having the formula (K->K%N K - no of pages, n - no of sets) the one with 2 pages will cause cache trashing when iterating through columns.

17. How many data blocks can be referenced to by the triple-indirection of an i-node, if a block contains N addresses to other blocks?

N^3

18. Consider the producer consumer problem with a buffer of capacity N. How many semaphores would you use to insure operation correctness and what would be the semaphores' initial values?

N semaphores with initial value 0

1. Why a hard link can be created only toward files on the same partition and not toward files on other partitions?

The **File system** is composed by a directory structure composed for directory entries to organize files. Each directory entry associates a file-name with an **inode**.

Soft links (**symbolic**) are directory entries that does not contain data, it just points to another entry (a file or directory in the same file system or other file system). And when you delete the pointed file, the symbolic link becomes unusable.

Hard links are directory entry that contains the file name and **inode** number. When you remove the last hard link, you can no longer access the file.

Hard-links are only valid within the same File system, because inodes are a data structure used to represent a file-system object, which are internal to the File system, and you can't point to an inode of another file system.

soft-links (Symbolic link) can span file systems as they simply point to another directory entry (The interface of the file-system, and not an internal object).

2. What is the principle of locality regarding page loading?

A process is likely to need soon the pages next to the page that was just loaded; predict what the program would need and when you load a page prefetch its neighbours

3. Schedule the following jobs so that so that the sum their delays is minimized: A/3/8 B/10/15 C/3/5

CAB (delay 1 sec)

4. What will the code fragment below print to the console?

```
char* s[3] = {“A”, “B”, “C”};
for(i=0; i<3; i++){
        execl(“/bin/echo”, “/bin/echo”, s[i], NULL);
}
```

RASP: A

5. Consider that function f is executed simultaneously by 10 threads. What would you add to insure that the value of n is 10 after the threads have completed? In this context what is line “n++” called, and what is variable n called?

```
pthread_mutex_t m;
int n = 0;
void* f(void *p){
```

```
        pthread_mutex_lock(&m);
        n++;
        pthread_mutex_unlock(&m);
        return NULL;
}
```

6. What are the elements of a virtual address in the paged-segmented allocation, and what tables are involved in calculating the physical address?

Elements of a virtual page: virrtaul pages and offset

Tables: RAM and the program(virtual address)

7. What does the system call “write” do when the fifo contains less space then the data it is required to write, but it is not full?

8. Give the operators for the following shell verifications:

- string is not empty -n
- is executable -x
- is different (for numbers) -ne
- is directory -d