[redacted]

Group: 9IT

Grade: 2.85

1. Write a UNIX Shell command that displays all the lines in file a.txt that contain at least one number with more than 2 decimal digits (i.e. -2.456, 4.234, 0.457).

grep "[0-9]*\.[0-9][0-9][0-9]*" a.txt

5

2. Write a UNIX Shell command that eliminates all non-letter characters from file a.txt (i.e. a B,5-9.+8 -> aB59)

sed "s/[^A-Za-z]//g" a.txt

4

3. Write an AWK program that applied to a file containing words separated by spaces, calculates the average word count per line.

awk '{wc=0; for (col=1; col<=NF; col++) wc++; count++; done} print $wc / $count}'

10

4. Display all the unique file names (without the path) in a given directory and all its hierarchy of subdirectories.

unique -u -f

5. Write a UNIX shell script that calculates the average number of lines in all the files with the .txt extension in the current directory.

count=0; lc=0;
for file in find *.txt
do lc += wc -l file;
count++;
done
print $lc/$count

4

6. How many processes will be created by the following fragment being executing the initial parent process?

for(i=0; i<4; i++)
fork();

15

7. How many processes are created by the parent process P, when it calls fork() and what is their relationship with each other and with the parent process P?

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

0

7, they are parallel processes and they are the child proc and they are the child process of the parent.

8. 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);
}
```

/bin/echo A
/bin/echo

A B C

9. What does the system call "read" do when the FIFO contains less data then it is required to read?

returns 0

10. What will the code fragment below print to the console, if no other process opens the "abc" FIFO? Justify your answer.

```
int r, w, n = 0;
r = open("abc", O_RDONLY);
n++;
w = open("abc", O_WRONLY);
n++;
printf("%d\n", n);
```

it will print 2 and it will have the value 0

11. What happens with a process between the moment it finishes and the moment its parent calls wait?

0

It runs the instructions or it executes.

12. Consider that function f is executed simultaneously by 10 threads. What would you add to ensure that the value of n is 10 after the threads have completed? In this context, what is line "error" called, and what is variable "n" called?

```
int n = 0;
void* f(void* p) {
    n++;
    return NULL;
}
```

5

the variable n is called global
we need a mutex_lock before n++
and a mutex_unlock after to prevent
the threads to increment for eg: 7 two times
and with only resulting one 8 → B written
twice.

13. Schedule the following jobs (given as Name/Duration/Deadline) so that the sum of their delays is minimized: A/7/13, B/5/9, C/2/4

10

CBA

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

5

adv: are much faster
dis: more memory

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

—

the segmentation

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 requests: 20, 9, 18, 27, 20, 9, 18, 27. Why?

9

the second one because it has more slots

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

3

N-1

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?

—

Semaphores will have the value 0 value 1
and we need N semaphores.

19. Give a method for preventing the apparition of deadlock.

—

We can avoid deadlocks by eliminating
the preemption

20. Add the necessary instructions to the code fragment below, so that the standard input of command /bin/pwd to be read from PIPE p.

2

```
int p[2];

pipe(p);

if(fork() == 0) {
    popen("/bin/pwd", "r", p[0]);
    execl("/bin/pwd", "/bin/pwd", NULL);
    pclose("/bin/pwd", p[0], p[1]);
    exit(0);

}
```