**1.** Scrieti o comanda UNIX care afiseaza toate liniile din fisierul a.txt care contin cel putin un numar binar multiplu de 4 cu cinci sau mai multe cifre (ex: 010100).

**grep -E '[0-1]{3,}[0]{2}' a.txt**

Explicatie: ultimele 2 cifre din fiecare nr binar trebuie sa fie 0 (ele reprezinta suma cu 2 si 1, daca ele sunt 1 atunci nu e multiplu de 4). In rest, cifrele din stanga pot fi in numar de 3 sau mai mare decat 3 (ca zice cu cinci sau mai multe cifre).

**2.** Scrieti o comanda UNIX care inverseaza toate perechile de cifra impara urmata de vocala

**echo a23e8i97u3 | sed -E 's/([13579])([aeiou])/\2\1/gi'**

Explicatie: primul caracter sa fie oricare din lista [13579], al doilea oricare din lista vocalelor, dupa dai swap global netinand cont de majuscule

3. Scrieti o comanda UNIX care afiseaza toate scorurile de fotbal unice (ex: 4-0) care apar in fisierul a.txt. Numarul de goluri poate avea maximum doua cifre

**cat a2.txt | sed 's/ /\n/g' | uniq -u | grep -E '^[0-9]{1,2}-[0-9]{1,2}$'**

Explicatia: inlocuiesti spatiul cu enter, ca sa ai fiecare scor pe o linie. folosesti uniq –u ca sa obtii doar liniile unice , apoi grep pentru a respecta conditiile.

**4**. Afisati numarul de procese ale fiecarui utilizator activ din sistem

**who | awk '{print $1}' | sort | uniq | awk 'BEGIN{ORS=" "}{print $1;system("ps -u "$1"|wc -l")}'**

**who | awk '{print $1}' | sort | uniq | awk '{print "ps -u "$1" | wc -l" | "/bin/sh"}'**

Explicatia: who pentru a afla utilizatorii activi , dai sot si uniq sa ii ai o singura data. La awk-> pui delimitatorul (ors) sa fie spatiu sa se vada frumos, apoi printezi utilizatorul si apoi dai comanda system ca sa afli cate procese are

**5**. Scrieti un script Shell UNIX care calculeaza media de fisiere cu extensia .txt per director din directorul curent si toate subdirectoarele lui.

#!/bin/bash

nr\_txt=0;

nr\_dir=1;

for f in $(find `pwd` -type f); do

if [[ $f == \*.txt ]]; then

echo $f is a .txt file;

nr\_txt=`expr $nr\_txt + 1`;

fi

done

for d in $(find `pwd` -type d); do

echo $d is a dir;

nr\_dir=`expr $nr\_dir + 1`;

done

echo There are $nr\_txt text files;

echo There are $nr\_dir directories;

**6**. Care procese va crea fragmentul de cod de mai jos, excluzand procesul parinte initial ?

if(fork()!=fork()){

fork();

}

=> 6 (7 cu parinte)

**8**. Ce tipareste in consola fragmentul de cod de mai jos ?

char\* s[3] = {"A", "B", "C"};

for(i=0; i<3; i++){

if(fork() != 0) {

execl("/bin/echo", "/bin/echo", s[i], NULL);

}

}

Se executa doar procesele parinte. Tipareste A,B,C (intr-o ordine random. ?)

**9**. Ce face apelul sistem “write” cand in PIPE este spatiu, dar nu suficient pentru cat i se cere sa scrie?

Uneori un write de succes poate transfera mai putini bytes decat numarul care s-a dat(count). Astfel de scrieri partiale pot aparea in diferite contexte, unul dintre ele fiind cel de sus din cerinta.

Scrie cati bytes are liberi si returneaza cati bytes a scris corect.

**10**. Ce tipareste fragmentul de cod de mai jos daca niciun alt proces nu deschide FIFO-ul “abc”? Justificati raspunsul.

int w, n, k=10, r;

r = open("abc", O\_WRONLY);

n = write(r, &k, sizeof(int));

printf("%d\n", n);

Nu va tipari nimic. Writing to a pipe or FIFO that doesn't have a reading process is treated as an error condition; it generates a SIGPIPE signal, and fails with error code EPIPE if the signal is handled or blocked.

**11**. Ce se intampla cu procesele zombie ale caror parinte s-au terminat?

Sistemul pastreaza procesul copil intr-o stare zombie, el nu executa nimic dar apare in lista de procese, ceea ce este problematic pentru ca pot creste in numar, ocupand PID uri and bringing the system to a point where no other processes can be created because of lack of PIDs. (vezi varianta root)

**12**. Considerati ca functia f este executata simultan de 10 thread-uri. Adaugati liniile de cod necesare ca sa asigurati ca n va avea valoarea 10 dupa ce thread-urile isi incheie executia ?

int n=0;

pthread\_mutex\_t mtx=PTHREAD\_MUTEX\_INITIALIZER;

void\* f(void\* p) {

pthread\_mutex\_lock(&mtx);

n++;

pthread\_mutex\_unlock(&mtx);

return NULL;

}

\*\* in main pthread\_mutex\_destroy(&mtx);

13. Planificati executia job-urilot urmatoare (date ca Nume/Durata/Termen) incat suma intarzierilor job-urilor sa fie minima: A/22/27, B/2/15, C/4/5

C, B, A

14. Dati un avantaj si un dezavantaj a cache-urilor set-asociative fata de cele directe.

Avantaj: direct-cache will cause cache collisions (cache trashing), and set-associative cache won’t

Dezavantaj: N comparators vs 1

15. Care este cea mai prioritara categorie de pagini de memorie din care politica de inlocuire NRU ar alege o pagina victima

The first victim of NRU will be the pages that have both bits r/w set to 0.

16. dupa primul for , lock la m[0] si m[2]

17. Dandu-se 2 cache-uri set-asociative, unul cu 2 seturi de 4pagini si unul cu 4seturi de 2pagini, care va da rezultatele mai bune pt secventa de cereri de pagini: 14,23,1,16,1,23,16,14. Justificati raspunsul.

4seturi de 2 pagini.

14%4 = 2

23 -> 3

1 -> 1

16 -> 0

1 -> 1

23 -> 3

16 -> 0

14%4 = 2

pt 4seturi de 2 pagini. Gr: 0 – 16, 1-1, 2-14, 3-23

pt 2seturi de 4 pagini . Gr: 0->14, 16 .. 1->23, 1

Depinde de cum merge:

18. Cate blocuri de date pot fi referite prin tripla-indirectare a unui i-node, daca un bloc are dimensiunea B si o adresa are dimensiunea A.

(B/A)^3

19. Ce se intampla cu un link hard cand fisierul spre care puncteaza este sters?

(fisier = inode care arata catre un bloc de date)

For a hard link, the i-nodes point towards the data, so when we delete a file, the data remains.

20. Dati o metoda pt prevenirea deadlockului in conditiile. .....

Same order (see cheatsheet)

1. Write a UNIX Shell command that displays the lines in a file a.txt that contains words starting with capital letters.

grep -E "^[A-Z]" a.txt

2. Write a UNIX Shell command that inverts in file a.txt all pairs of neighboring digits

sed -E "s/([0-9])([0-9])/\2\1/gi" a.txt

3. File a.txt contains on each line two numbers separates by space. Write a UNIX Shell command that displays for each line the sum of its numbers.

awk '{print s=$1 + $2 }' number

4. Display only the lines of file a.txt that appear only once (not duplicated)

cat a.txt | sort | 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”

files=`find ./ -maxdepth 1 -type f`

ok=0

for file in $files

do

IFS=$'\n'

ok=0

for line in ` cat $file `

do

if echo $line | grep -E -q "cat"

then

ok=1

fi

done

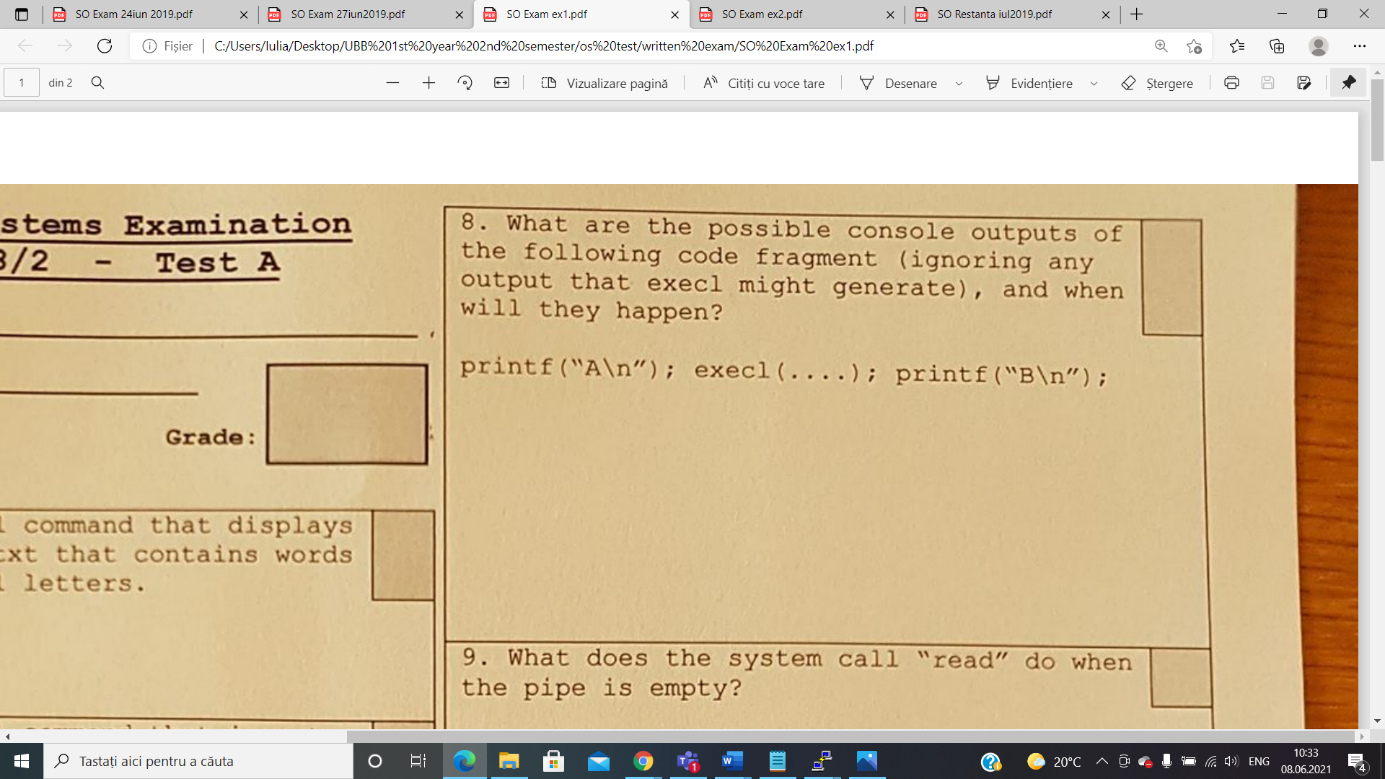
if [ $ok -eq 1 ]

then

echo $file

fi

done



Case 1: (execl succeeds)

* The output will be A because execl commands wipes the initial code and replace it with the code of the new program

Case 2: (execl does not succeeds)

* The output will be A \*newline\* B because the execl call failed so the calling process continues to execute

9. What does the system call “read” do when the pipe is empty

The system call will wait for data if the is a writer connected or the pipe will see the end of file (read will return 0)

10. What does the system call “open” do before returning from opening a FIFO?

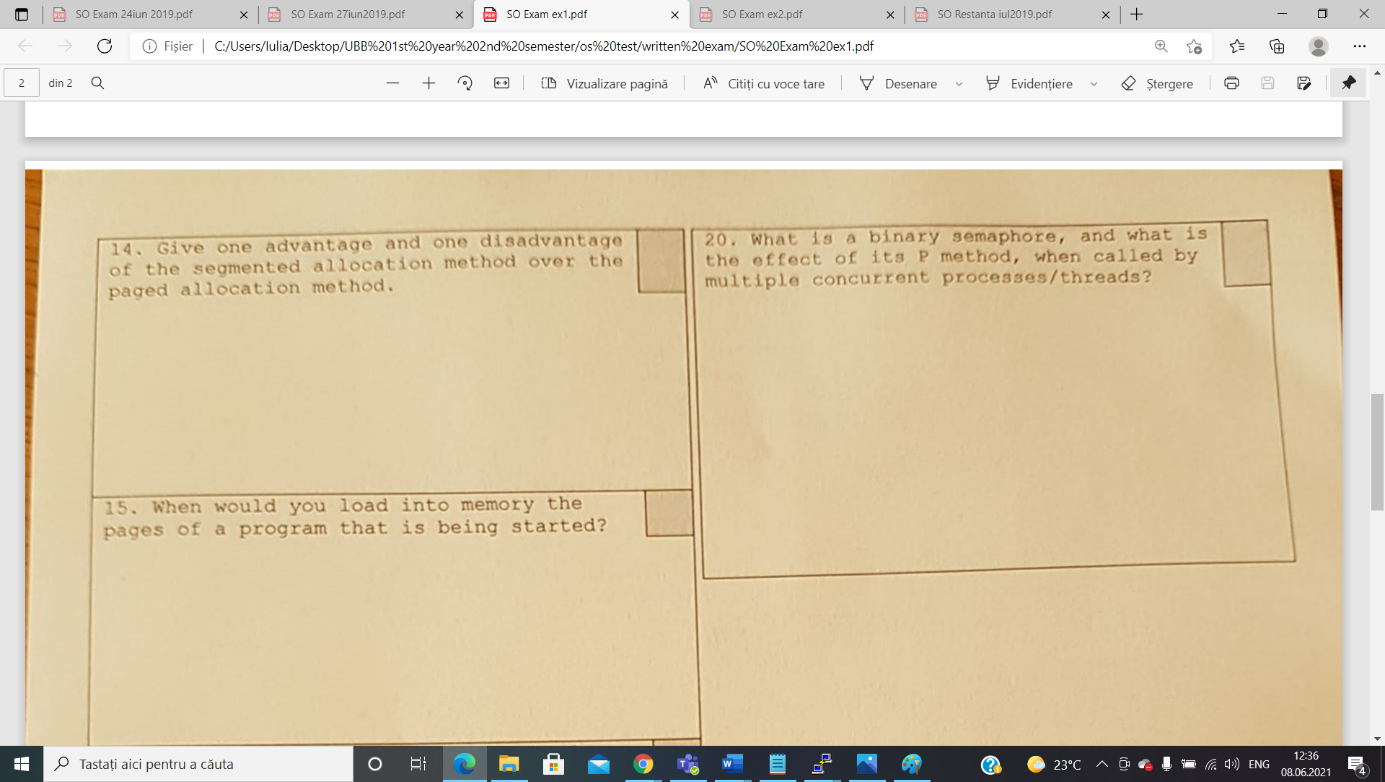
The system call will wait until the complementary operation in the other process is opened.

11. Give a reason for choosing threads over processes

Sometimes it’s better to use threads instead of processes because threads use less resources taking into consideration that they have a shared memory, they use global variables etc, whereas processes does not share data, consume more resources and takes more time for creation and communication between them.

12. The result is uncertain because there is no efficient synchronization between the threads as there are 2 global mutexes which are used in two different functions instead of one used globally in both of them.

13. A/5/9; C/1/10; B/7/13

14. 

Disadvantage:

* The segmented allocation method is slower than the paging one, in terms of memory access

Advantage:

* The segmented allocation method has a better protection of memory because each segment may receive other access rights.

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 just like a mutex lock, it has 2 values( 0 and 1) and it is initialised with 1. Concurrent threads will wait at the binary semaphore if its value is 0.  A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1)wait, and 2) signal for the process synchronization.

The P method makes concurrent threads wait at the semaphore if its value is 0 (P=wait, sleep, down operation)

1. Avantaj si dezavantaj al alocarii cu partitii variablie si alocarea paginata

* **Avantaj**: Considering the fact that the address doesn’t need to be translated from a virtual address to a physical address we can say that partitioned allocation is faster from a computational point of view
* **Dezavantaj**: Apare fenomenul de fragmentare internA a memoriei deoarece procesele aloca memorie iar apoi sunt eliberate, Ramanand spatii libere care nu vor fi ocupate deoarece dimensiune lor va scadea.



The most prioritary class of pages that would be chosen as a victim by the NRU policy is the one that was least recently used.



The system call “open” will wait for the complementary operation (write) to be opened in another process.

-> {x,y } x- minimum number of times we want to repeat; y- maximum number of times we want to repeat

-> we repeat (write) the regular expression “n” times

-> \* zero or more times

-> + one or more times

-> ? 0 or 1 time

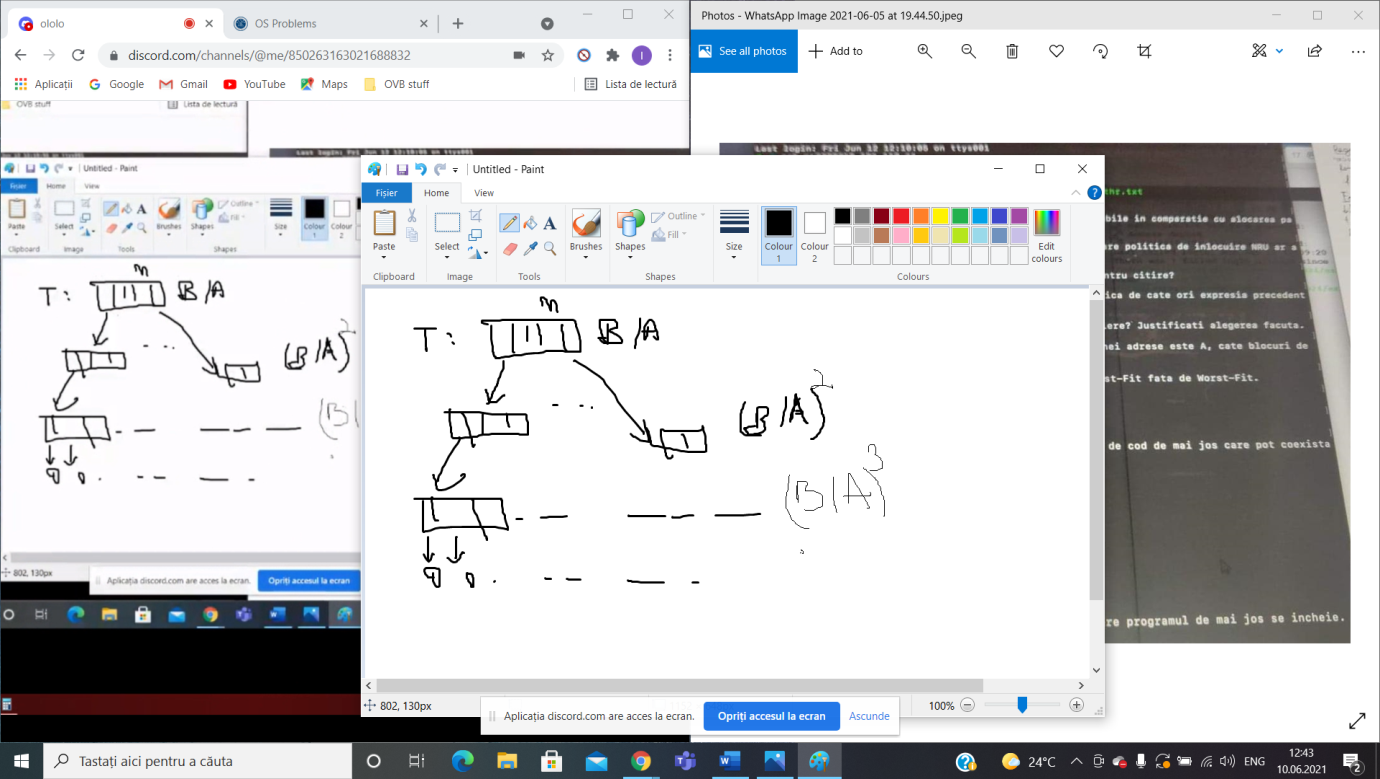
5. Cate threaduri ati folosi

-> We will use one million threads because in case of using I/O operations we can benefit from using a number of threads grater than the number of cores of the CPU because threads will enter in wait state when it comes to I/O (files) operations so while they are working on the hard disk

other threads can work on other stuff.

1. **Considerand ca dimensiunea unui bloc este B si dimensiunea unui adrese este A , cate blocuri de date sunt adresate de indirectarea tripla a unui i-nod?**

(B/A)^3



1. FIRST FIT vs WORST FIT

* AVANTAJ: First fit is faster.
* DEZAVANTAJ: There might be some small pieces of memory where it’s difficult to store other data.

1. **Cand trece un process din starea de RUN in starea de READY**

* A process will change its state from RUN to READY after he did everything he had to do on his allocated time on the processor (after its time on processor expires).