SUB1 - 2023

1. **Dati 3 expresii regulare care accepta orice nr ne-negativ multiplu de 5 .**

^(0| [1-9]\*0| [][1-9]\*5)$

^(0| [][1-9]\*[05])$

^(0| [][1-9]\*(0|5))$

1. **Dati 5 comenzi GREP care afiseaza toate linile dintr-un fisier care contin litera “a”/”A”**

grep -E ‘[Aa]’ file

grep -i ‘a’ file

grep -E ‘a|A’ file

grep -E ‘a|A{1,}’ file

grep -E ‘a|A+’ file

1. **Scrieti doua comenzi SED care afiseaza dintr-un fisier doar linile care nu contin cifra 7**

sed -E ‘/7/d’ file

sed -E -n ‘/7/!p’ file

1. **Scrieti o comandä AWK care afiseazà suma penultimului câmp al tuturor liniilor.**

awk ‘BEGIN {sum = 0}{sum = sum + $(NF-1)} END {print sum}’

1. **Cum puteti redirecta in linia de comanda iesirea de eroare prin pipe inspre un alt program?**

comanda 2>&1 | alt\_program

(2>&1 redirectioneaza iesirea de eroare stderr catre iesirea standard stdout)

1. **Scrieti un script Shell UNIX care afiseazà toate argumentele din linia de comanda fára a folosi FOR.**

#!/bin/bash

while [$# -gt 0]; do

echo $0

shift 1

done

1. **Desenati jerarhia proceselor create de codul de mai jos, incluzand procesul parinte.**

**for (i=0; 1<3; i++) {**

**fork();**

**execlp("ls", "ls", "/", NULL);**

**}**

* Daca exec nu da fail: P->C1 (codul va fi inlocuit cu executia exec si nu se vor reintorce in cel principal)
* Daca exec da fail: P->C1->C2->C3 (executia continua in programul principal)

1. **Adäugati codul C necesar pentru ca fisierul b.txt sà fie suprascris cu continutul fisierului a.txt din instructiunea de mai jos. -> execlp("cat", "cat","a. txt", NULL);**

int fd = open(“b.txt”, O\_CREAT | O\_WRONLY);

dup2(fd, 1);

execlp("cat", "cat","a. txt", NULL);

1. **De ce nu e recomandat sa comunicati bidirectional printr-un singur FIFO?**

* Nu e recomandata folosirea unui singur FIFO pentru comunicarea bidirectionala deoarece acel fifo va permite citire/scriere pentru ambele procese iar ordinea de scriere nu se cunoaste cee ace poate duce la coruperea datelor.

1. **Câte FIFO-uri poate deschide un process dacã nu sunt si nici nu vor fi folosite vreodatä de vreun alt proces?**

* Daca deschidem unul sau mai multe FIFO-uri fara a le folosi intr-un alt process procesul initial va fi blocat deoarece functia de open/read va astepta open/read din celelalt capat/de la celelate procese si nu l va primi

1. **Când ati folosi un process in locul unui thread?**

* Un process ar trebui folosit in locul unui thread atunci cand tinem cont de securitatea unui program. Fiecare process are un spatiu propriu de adrese de memorie (o eroare intr-un process nu influenteaza mersul celorlalte procese) , in timp ce threadurile lucreaza pe o memorie partajata.

1. **Ce este o "sectiune criticã"?**

* O sectiune critica este o portiune de cod care acceseaza resurse partajate care nu trebuie accesate simultan de mai multe fire de executie sau procese. Scopul unei sectiuni critice este de a prevenii conditiile de cursa.

1. **De ce un thread trebuie sã reverifice conditia la iesirea din apelul pthread \_cond \_wait?**

* Un thread ar trebui sa reverifice conditia la iesirea din apelul cond\_wait pt ca alte threaduri ar putea sa astepte in aceeasi variabila conditionala, ar putea modifica starea si semnaliza conditia inaintea lui

1. **Care va fi efectul inlocuirii apelurilor la pthread\_mutex\_lock cu apeluri la pthread\_rwlock\_rdlock?**

* Inclocuirea apelurilor mutex\_lock cu rwlock\_rdlock are effect asupra concurentei si accesului la resursele partajate. Un mutex permite accesul unui singur thread care poate realiza or citire or scriere, in timp ce un rwlock permite multiple citiri concurente, blocate simultan (rdlock), ceea ce poate fi mai efficient in cazul in care avem un numar mai mare de citiroii decat de scriitori, dar daca nr de scriitori este mai mare aceasta metoda este ineficienta deoarece acestia vor astepta pt finalizarea fiecarui reader

1. **Care e efectul apelului la pthread\_barrier\_wait pentru o barierà initializatã cu 1?**

* Nu exista niciun efect dupa apelul pthread\_barrier\_wait initializata cu 1, pt ca o data ce apelam wait bariera v-a decrementa la 0 si va da voie thredului sa treaca- acelasi effect ar avea programul si daca pthread\_barrier\_wait nu ar fi apelat

1. **Cum puteti incrementa valoarea unui semafor?**

* Valoarea unui semafor se poate incrementa cu functia sem\_post

1. **Ce puteti face ca programator pentru a preveni deadlock-urile?**

* O solutie simpla pentru rezolvarea deadlockurilor este alegerea unei ordini pentru blocarea resurselor si pastrarea acelei ordini de a lungul procesului

1. **Prin ce tranzitie de stare va trece un process când citeste dintr-un fisier?**

Procesul are urmatoarele stari:

* READY + HOLD: procesul se afla in starea de pregatire, dar nu este inca un processor asa ca asteapta sa fie planificat pentru a rula
* RUNNING : procesul este in executie si efectueaza citirea din fisier
* WAIT: procesul astapta dupa ceva (ex: mutex), nu primeste procesorul pana nu este gata
* SWAP: cand se umple memoria e posibil ca procesul sa mai ceara un malloc asa ca pune memoria pe hard

1. **Ce contine superblocul unui disc Linux?**

* Superblocul contine informatii critice despre sistemul de fisiere, cum ar fi: identificatorul, starea, dimensiunea, unde incepe fieacre partitie, so creator al sistemului

1. **Se poate crea un link hard spre un fisier aflat pe o altà partitie? Justificati raspunsul.**

* NU se poate crea un hard link spre un fisier dintr-o alta partitie pt ca un hard link urmareste i-node a unui fisier. Avand in vedere ca i-node urile unui fisier sunt valabile doar in interiorul unei aceleasi partitii, nu am avea acces la i-node urile unui fisier dintr-o alta partitie, deci nu putem avea un hard link catre o alta partitie

SUB2 – 29.06.2023

1. **scrieti un grep care ia grupurile de cate 2 cuvinte, separate de un singur spatiu, care sunt formate doar din litere mici si fiecare cuvant contine cel putin 2 vocale**

grep "</[a-z]\*[aeiou][a-z]\*[aeiou][a-z]\*/> </[a-z]\*[aeiou][a-z]\*[aeiou][a-z]\*/>" fisier

1. **scrieti 2 grep uri care iau liniile care nu au numarul de caractere multiplu al lui 3**

grep -E -v "^(..)\*.$" fisier

1. **scrieti un sed care inlocuieste prima aparitie a caracterului A cu caracterul B**

sed -E "s/(^[^A]\*)(A)(.\*$)/\1B\3/gi" a.txt

sed -E "s/A/B/i"

**4. scrieti un awk care afiseaza liniile care au primul cuvant identic cu ultimul cuvant si al caror penultim cuvant are numar par de caractere**

awk 'NF > 1 && $1 == $NF && length($(NF - 1)) % 2 == 0 {print $0}'

**5. scrieti 3 moduri de a crea un fisier gol**

touch a.txt

echo "" > a.txt

echo 1> a.txt



**6. scrieti 5 moduri de a verifica daca un string este gol(cu test)**

- if test -z $A

- if test $A=""

- if test ${#A} -eq 0

- if test ${#A} -lt 1

- if test ! ${#A} -gt 0

- if test $(echo $A | awk '{print length}') -eq 0



**7. afisati ierarhia proceselor a urmatorului cod:**

**for(int i = 0; i < 3; i++)**

**if (fork() != 0)**

**wait();**

**8. scrieti ce afiseaza codul:**

**execlp(“expr”,”expr”,”a”,”+”,”1”);**

**printf(“xyz\n”);**

execlp da eroare (ultimul argument nu este NULL)

**10. cate FIFO pot fi deschise de catre un fisier daca fiecare dintre acele FIFO-uri va avea capatul celalalt deschis de catre un alt proces?**

- pot fi deschise oricate FIFO uri atat timp cat fiecare FIFO are capatul celelat deschis de catre un alt process

**11. cand am dori sa folosim execl si cand am dori sa folosim execv?**

- diferenta dintre cele doua este ca execv primeste ca argument un array in timp ce execl primeste o lista. Asa ca daca avem o comanda cu mai putine argumente putem folosi execl pt a nu ocupa memorie, insa daca comanda este compusa din multe argumente este mai avantajos si rapid de folosit execv

**14. care sunt consecintele inlocuirii lui pthread\_mutex\_lock cu sem\_post in cod?**

- diferenta majora este ca pthread\_mutex\_lock blocheaza accesul la o resursa comuna intre threaduri, in timp ce sem\_post este folosit pentru incrementarea unui semafor si nu blocheaza executia threadurilor. Deoarece nu va exista o bariera de protective pentru accesul simultan se va ajunge la condtii de cursa

**15. definiti un semafor binar si explicati cum functioneaza**

- un semafor binar este un semafor initializat cu 1 ce se comporta in esenta ca un mutex fiind folosit pentru eliberarea unei zone de cod incarcata

**18. prin ce stare trece un proces cand apelam pthread\_join?**

- cand apelam pthread\_join, procesul trece in stare de BLOCK+WAIT. Procesul care a apelat pthread\_join este blocat pana cand procesul specificat se termina. Apoi se asteapta terminarea executiei unui fir de executie ca resursele sale sa fie eliberate, iar informatiile de dupa terminare colectate (de asta “join”).

**19. daca avem B drept block size si A drept address size, cate adrese o sa aiba un double indirect dintr-un i-node?**

- double indirect = un bloc care contine adrese catre un bloc care, la randul lor, contin adrese catre blocuri de date

Un simple bloc contine B/A adrese, deci unul dublu indirect contine (B/A)^2 adrese

**20.ce se intampla cu continutul directorului in care montam o partitie?**

**-** montare constă în conectarea unui sistem de fişiere, de pe un anumit disc, la un director existent pe sistemul de fişiere implic

In momentul in care montam o partitie intr-un director existent, continutul original al acestui director (fisierele si subdirectoarele) va fi temporar acoperit de partitia montata. Continutul nu este sters ci doar “ascuns”. Acesta devine vizibil atunci cand demontam partitia

SUB3 – 2023 restanta

1. **Daţi o expresie regulară care acceptă orice secvenţă de lungime pară de cuvinte (formate din litere) separate prin spaţii, cu condiţia ca pentru fiecare cuvânt lungimea şi poziţia în secvenţă să fie ambele pare sau impare. Numărarea cuvintelor începe de la 1. Ex: al 5-lea cuvânt trebuie să aibă lungime impară, iar al 16-lea să aibă lungime pară.**

'\<([a-z]{2})+\> \<(([a-z]{3})+|[a-z])\>|\<(([a-z]{3})+|[a-z])\> \<([a-z]{2})+\>'

1. **Daţi trei comenzi GREP care afişează dintr-un fişier liniile formate exclusiv dintr-o secvenţă nevidă de litere şi cifre alternativ (ex: a0g sau 1r5m).**

grep -E -i '^(([a-z]([0-9]|[0-9][a-z]))+|([0-9]([a-z]|[0-9]))+)$' fisier

grep -E '^[0-9]?([a-zA-Z][0-9])+[a-zA-Z]?$' fisier

grep -E -i '^[a-z]?([0-9][a-z])+[0-9]?$' fisier

1. **Scrieţi două comenzi SED care afişează liniile unui fişier ştergând prima secvenţă nevidă de litere mici.**

sed '/^[a-z]+$/d’ fisier

sed '/^[a-z]{1,}$/d’ fisier

1. **Scrieţi o comandă AWK care afişează suma tuturor numerelor dintr-un fişier text ale cărui linii sunt formate din secvenţe de cifre separate prin spaţii.**

awk 'BEGIN {sum = 0}{for(i = 0; i<NF; i++) sum= sum + $i} END {print sum}'

1. **Daţi trei moduri de a afla dimensiunea unui fişier în linia de comandă Linux.**
2. ls -l fisier (se extrage nr de octeti de pe a treia coloanal de la dr)
3. du -h fisier (afiseaza dim in KB)
4. stat fisier (afiseaza si dimensiunea pe langa alte detalii)
5. **Scrieţi o condiţie Shell UNIX care verifică dacă un un fişier există şi utilizatorul curent are vreo permisiune asupra lui.**

if [ -e file] && [ -r file -o -w file -o -x file]

1. **Desenati ierarhia proceselor create de codul e mai jos, incluzand procesul parinte.**

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

**if(fork() > 0) {**

**wait(0);**

**wait(0);**

**exit(0);**

**}**

**}**

P -> C -> C -> C

1. **Ce va tipări fragmentul de cod de mai jos? Justificaţi răspunsul.**

**execl("expr", "expr", "1", "+", "1", NULL);** - daca reuseste se va afisa 2, dc nu trecem mai departe

**execlp("echo", "echo", "3", NULL);** - dc reuseste se va tipari 3, dc nu se trece mai departe

**printf("4\n");** - dc exec dau fail se afiseaza 4

1. **De ce descriptorul de fişier returnat de popen trebuie închis cu pclose şi nu cu fclose?**

* Functia popen este folosita pentru a creea un process si a deschide un flux de date de la/catre acel process. Functia returneaza un pointer la un file asociat cu acel process. Pclose nu doar inchide fluxul de date, ci si asteapta terminarea porcesului copil, spre deosebire de fclose (unde porcesul copil poate devein “zombie”)

1. **Câte FIFO-uri poate deschide pentru citire un process, dacă FIFO-urile sunt şi vor fi întotdeauna folosite de alte procese doar pentru scriere?**

* Un process poate sa deschida oricat de multe FIFO-uri pentru citire, avand asigurarea ca procesul care il foloseste il v-a deschide pentru scriere

1. **Daţi trei apeluri de funcţii care asigură excludere mutuală.**

pthread\_mutex\_lock

pthread\_mutex\_unlock

sem\_wait (pentru un semafor binar)

1. **Ce se poate întâmpla dacă funcţia f este rulată de mai multe thread-uri simultane? De ce?**

**pthread\_mutex\_t m[2];**

**void\* f(void\* p) {**

**int id = (int)p;**

**pthread\_mutex\_t\* first = &m[id % 2];**

**pthread\_mutex\_t\* second = &m[(id+1) % 2];**

**pthread\_mutex\_lock(first);**

**pthread\_mutex\_lock(second);**

**...**

**pthread\_mutex\_unlock(second);**

**pthread\_mutex\_unlock(first);**

**}**

* Rularea cu mai multe threaduri poate duce la deadlock (blocari reciproce). De exemplu daca un thread cu p = 0 apeleaza functia acesta v a bloca mutexul m[0] si v a incerca sa blocheze m[1], si simultan un thread cu p = 1 apeleaza si el functia f, acesta v a bloca m[1] si va incerca sa blocheze m[0] => deadlock intre cele doua threaduri
* De asemenea unlock ul ar trebui facut in ac ordine cu lock-ul pentru corectitudine si respectarea prioritatii

1. **Daţi un exemplu de valori distincte şi mai mari ca 0 pentru T, N1, N2 şi N3 pentru care programul de mai jos se incheie.**

**pthread\_barrier\_t b1, b2;**

**void\* f1(void\* a) {**

**pthread\_barrier\_wait(&b1);**

**return NULL;**

**}**

**void\* f2(void\* a) {**

**pthread\_barrier\_wait(&b2);**

**return NULL;**

**}**

**int main() {**

**int i;**

**pthread\_t t[T][2];**

**pthread\_barrier\_init(&b1, NULL, N1);**

**pthread\_barrier\_init(&b2, NULL, N2);**

**for(i=0; i<T; i++) {**

**pthread\_create(&t[i][0], NULL, f1, NULL);**

**pthread\_create(&t[i][1], NULL, f2, NULL);**

**}**

**for(i=0; i<T; i++) {**

**pthread\_join(t[i][0], NULL);**

**pthread\_join(t[i][1], NULL);**

**}**

**pthread\_barrier\_destroy(&b1);**

**pthread\_barrier\_destroy(&b2);**

**return NULL;**

**}**

1. **Prin ce tranziţie de stare va trece un process când apelează sem\_wait şi în ce condiţii? Justificaţi răspunsul.**

* Daca semaforul are o val > 0 acesta decrementeaza val si trece mai departe continuand executia, daca valoarea este < 0, procesul trece in starea de WAIT active adica, astepta sa primeasca un semafor verificand constant starea acestuia, iar daca val = 0 intra in starea de BLOC unde asteapta ca val semaforului sa devina mai mica decat 0

1. **Cand ati incarca in memorie paginile unui program care tocmai este pornit?**

Putem sa incarcam toate paginile de la bun inceput (dezavantaje: pornire lenta, memorie ocupata de pagini care nu se vor folosi; avantaje: odata incarcat, merge repede) sau incarcam fiecare pagina cand devine necesar (dezavantaje: rulare mai lenta, avantaje: pornire rapida, paginile nefolosite nu ajung in memorie)

**SUB 4 – 2023**

**1.RO: Daţi o expresie regulară care acceptă orice număr impar de cuvinte separate prin spaţii, fiecare cuvânt având număr impar de litere**

**2. RO: Daţi patru comenzi care afişează numărul de linii goale dintr-un fişier.**

grep -E -c "^$" a.txt

grep -E -V -c "..\*" a.txt

grep -E -v -c ".+" a.txt

grep -E -v -c ".{1,}" a.txt

**3. RO: Scrieţi o comandă SED care afişează liniile dintr-un fişier ştergând din ele primul, al treilea, al cincilea, al şaptelea, etc spaţii.**

sed -E "s/( )([^ ]+)?( )?/\2\3/gi" file

**4. RO: Scrieţi o comandă AWK care afişează produsul ultimului câmp al liniilor de pe poziţii impare care au număr impar de câmpuri.**

awk 'BEGIN { p = 1} { if (NF % 2 == 1 && NR % 2 == 1) p \*= $NF} END { print p}' fisier

**5. RO: Daţi patru moduri prin care ieşirea standard a unui proces poate fi redirectată.**

1) >

2) pipe |

3) functii dup si dup2

4) >>

**6. RO: Scrieţi trei condiţii Shell UNIX care verifică existenţa unui fişier. - 8,**

1) if [ -e $file ]

2) if [ $(ls $file 2>&1 | grep -E -c "No such file") -eq 0 ]

3) if [-f $file ]

**7. RO: Desenati ierarhia proceselor create de codul de mai jos, incluzand procesul parinte.**

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

**if(execlp("ls", "ls", "/", NULL) != -1) {**

**fork();**

**}**

**}**

P->C1

**8. RO: Adăugaţi codul C necesar pentru ca instrucţiunea de mai jos să nu se blocheze aşteptând la intrarea standard. execlp("cat", NULL);**

int fd = open("/dev/null", O\_RDONLY);

dup2(fd, STDIN\_FILENO);

execlp("cat", "cat", NULL);

**9. RO: Schiţaţi o implementare a funcţiilor popen şi pclose, doar pentru cazul în care outputul comenzii e citit în codul C**

FILE\* mypopen(char\* cmd, char\* type) {

int p[2], caller\_idx, child\_idx;

pipe(p);

caller\_idx = 0;

if(type[0] == 'w') {

caller\_idx = 1; }

child\_idx = (caller\_idx+1) % 2;

if(fork() == 0) {

close(p[caller\_idx]);

dup2(p[child\_idx], child\_idx);

if(execlp("bash", "bash", "-c", cmd, NULL) < 0) {

close(p[child\_idx]);

exit(1); } }

close(p[child\_idx]);

return fdopen(p[caller\_idx], type); }

void mypclose(FILE\* fd) {

fclose(fd);

wait(0); }

// Usage example

FILE\* f=mypopen("who", "r"); // read from f

mypclose(f);

FILE\* f=mypopen("less","w"); // write to f

mypclose(f);

**10. RO: Câte FIFO-uri poate deschide pentru citire un process, dacă FIFO-urile sunt şi vor fi întotdeauna folosite de alte procese doar pentru citire?**

- nu se poate deschide niciun FIFO pentru citire intr -un process atat timp cat nu exista al process car esa il deschids pentru scriere

**11. RO: Când aţi folosi un FIFO în locul unui pipe?**

- FIFO sunt utile pentru comunicarea intre procesele ce nu au o relatie parinte-copil si care nu sunt lansate in acelasi timp. De asemenea FIFO urile pot fi create o data si folosite de mai multe ori, chiar daca procesul care le a creat isi termina executia atat timp cat nu sunt sterse, in timp ce pipe-urile dispar automat.

**13. RO: Când aţi folosi un mutex în locul unui rwlock?**

- Folosirea unui mutex în locul unui rwlock este recomandată atunci când accesul exclusiv la o resursă este necesar sau când operațiunile de scriere sunt preponderente. În schimb, rwlock-urile sunt utile atunci când accesul concurent pentru citiri este esențial și operațiunile de citire sunt mult mai frecvente decât cele de scriere..

**14. RO: Care va fi efectul înlocuirii apelurilor la pthread\_mutex\_lock cu apeluri la sem\_wait?**

**-** mutex\_lock asigura excludere mutuala, doar un thread detine mutexul, iar daca semaforul este initializat cu o valoare mai mare decat 1 mai multe threaduri vor avea acces in ac timp in sectiunea critica. In schimb daca semaforul este initializat cu 1, sem\_wait ar putea avea ac comportament ca sip thread\_mutex\_lock

**15. RO: Ce face pthread\_cond\_wait cu mutex-ul primit ca argument?**

- in momentul apelului mutexul se deblocheaza (UNLOCK) (previne race conditions), mai apoi asteapta semnalarea si cand o primeste de la un alt thread se blocheaza (LOCK)

**16. RO: Schiţaţi o soluţie pentru problema producător-consumator.**

*semaphore plin; - indica nr de pozitii ocupate din buffer*

*semaphore gol; - indica nr de pozitii libere din buffer*

*semaphore exclus; - ofera accesul exclusive la buffer*

*v0(plin) = 0;*

*v0(gol) = n;*

*v(exclus) = 1; - semafor binar / mutex*

*PRODUCATOR:*

*do {*

*<produce articol>;*

*P(gol);*

*P(exclus);*

*<depune articol în buffer>;*

*V(exclus);*

*V(plin);*

*} while(false);*

*CONSUMATOR*

*do {*

*P(plin);*

*P(exclus);*

*<extrage articol din buffer>;*

*V(exclus);*

*V(gol)*

*<consuma articol>;*

*} while(false);*

**18. RO: Prin ce tranziţie de stare va trece un process când apelează pthread\_cond\_wait? Justificaţi răspunsul.**

- RUN to BLOCK/WAIT : deblockeaza mutexul in timp ce asteapta semnalarea conditiei

**19. RO: Ce conţinue un fişier de tip director în sistemul de fişiere Linux?**

- lista de perechi formate din nume si i-node pentru fisierele subordonate

1. **RO: Explicaţi diferenţa dintre un link simbolic şi un link hard.**

Un link hard este o legatura ce puncteaza spre o substructura din sistemul de fisier la care puncteaza deja legatura naturala de aceea pot fi făcute numai în interiorul aceluiaşi sistem de fişiere. Legăturile simbolice sunt intrări speciale într-un director, care punctează (referă) un fişier (sau director) oarecare în structura de directori

**Sau**

Legăturile hard funcţionează numai în interiorul aceluiaşi sistem de fişiere, în timp ce legăturile simbolice pot puncta şi spre noduri ale altui sistem de fişiere montat împreună cu sistemul de fişiere ce conţine legătura limbolică.

**SUB5 – 2019**

**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 ( cinci sau mai multe cifre).

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

sed -E 's/([13579])([aeiou])/\2\1/gi' fisier

**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 ‘{print $1; system("ps -u "$1"| wc -l")}'

**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. Cate 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) {** - suntem in parinte

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

**}**

**}**

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?**

- asteapta catva spatiu sau pana nu mai sunt cititori, returneaza numarul de bytes scrisi sau 0 cu O\_NDELAY

**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 se va tipari nimic deoarece procesul este blocat dupa linia a doua, daca niciun alt process nu deschide FIFO ul in modul citire 0\_RDONLY

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

Sistemul pastreaza procesul copil intr-o stare zombie, atunci cand el se termina inainte ca parintele sa faca wait. El nu executa nimic dar apare in lista de procese, ceea ce este problematic pentru ca pot creste in numar, ocupand PID uri si facand imposibila crearea de alte procese atunci cand toate PID-urile ajung a fi ocupate.

**12. Considerati ca functia f este executata simultan de 10 thread-uri. Adaugati liniile de cod necesare ca sa asigurati ca 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);**

if(n<10)

**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**

Luam toate permutarile posibile: ABC, ACB, BAC, BCA, CAB, CBA si calculam intarzierile

* ABC : A termina la 22, intarziere 22-27 = -5 (nu intarzie)

B termina la 22+2 intarziere 24-15 = 9

C termina la 24+4 intarziere 28-5 = 23 => 23+9 = 32 intazriere pt ABC

* ACB : 11 + 23 = 34
* BAC : 13
* BCA : 1 + 1 = 2
* CAB : 13
* CBA : 1
* C, B, A

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

Avantaj: -urile set-associative reduc riscul de coleziuni (cache trashing)

Dezavantaj: sunt mai complexe si mai costisitoare de implementat (necesita o logica suplimentara pentru gestionarea mia multor blocuri per set)

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

Prima pagina cea mai prioritara in politica NRU pentru alegerea unei pagini victime este cea din clasa 0 (are ambii biti 0 si cel pt R si cel pt W) – este neatinsa

1. **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.**

- cacheul cu 2 seturi si 4 pagini, impartim astfle: set 0 – pagini cu adrese pare (14, 16)

set 1 – pagini cu adrese impare (23, 1)

maparea: set 0 – 14 (miss), 16(miss), 16(hit), 14(hit)

set 1 – 23 (miss), 1 (miss), 1(hit), 23(hit)

- cacheul cu 4 seturi si 2 pagini, impartim astfel: set 0 – adrese care dau restul 0 impartite la 4

set 1 - …

set 2 - …

set 3 - …

mapare: set 0 – 16(miss), 16(hit)

set 1 – 1(miss), 1(hit)

set 2 – 14(miss), 14(hit)

set 3 - 23(miss), 23(hit)

=> amble au dat acelati rezultat: 4 cache hits si 4 cache misses => aceeasi performata

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

Pentru un hard link, i-node ul pointeaza catre data, asa ca atunci cand stergem un fisier, data ramane.

SUB6 – 2015

1. **Write a sed command which duplicates the numbers in a file and prints it to the output**

sed -E 's/([0-9]+)/\1\1/g' fileinput > fileoutput

**2. Print the lines from a.txt which ends in vowels**

grep -E -i '[aeiou]$' a.txt

**3. In a.txt each row has 3 numbers separated by ":". Write an awk script which prints the sum of the numbers from the 2nd position**

Awk ‘BEGIN {FS = “:”, sum = 0} {sum = sum + $2} {print sum}’ a.txt

**4. Print the path of the current, parents and root directory**

current: pwd

parents: cd .. ; pwd

root: cd / ; pwd

1. **How many processes will be created? fork(); fork(); fork(); wait(); wait(); wait(); (dont count the root process)**

2^3-1 = 7

**6. Write a shell script which prints all its arguments which are not files or directories. (No valiadation needed)**

#!/bin/bash

for arg in \*$; do

if [ ! -f $arg && ! -d $arg]; then

echo $arg

fi

done

**7. Write a shell script which takes 2 numbers as arguments and prints the greater one (No valiadtion needed)**

#!/bin/bash

if [ $1 -gt $2 ]; then

echo $1

else

echo $2

fi

**8. A shell script takes as argument the numbers "1 2 3 4 5". Write a script, which prints the sum of the numbers from the odd positions.**

#!/bin/bash

cnt = 0

sum = 0

for arg in \*$; do

cnt = $((cnt+1))

if [ $((cnt%2)) -eq 0 ]; then

sum = $((sum + $arg))

fi

done

**9. Explain why the parent process can't use a pipe, which has been created in the child process.**

- toate variabilele din copil sunt independente si nu pot fi accesate de catre parinte pt ca file descriptorul copilului nu e valabil pt parinte

**10. What is the logical value of a shell script which terminates with return value of 5**

- false (a returnat 5 pentru ca a aparut o eroare)

**11. Explain a case when the state of a process is changed from RUN to SWAP**

- cand procesul are mai multe procese ce ruleaza simultan si nu mai are memorie trece in SWAP

**12. Explain the roles of V and P in Semaphores. (short answers)**

- V: asteapta un semnal, decrementeaza val semaforului cu 1

- P: da un semnal ca o resursa e valabila, incrementeaza val semaforului cu 1

**14. Add 2 examples of the 3 possible, real memory allocation techniques**

- cu partitii fixe: absolute si reaclocabile, cu partitii variabile

**15. Which unix c function creates a new pipe**

pipe()

**16. What happens when we run the next commands at the same time.**

**(Note that fifos are created and can be used):**

**open("a2b", O\_RDONLY); open("b2a",O\_RDWR);**

**open("b2a", O\_RDONLY); open("a2b",O\_RDWR);**

* Ambele FIFO uri asetapta unul dupa altu si dac anu exista o ordine se intampla deadlock

SUB7 – 2015

1. **Sa se scrie o comanda UNIX care afiseaza liniile care au numarul total de caractere divizibil cu 3.**

grep -E '^([^ ]{3})+$'

**2. Sa se scrie o comanda UNIX care duplica primele 5 caractere de pe fiecare linie ex: abc01xyz => abc01abc01xyz**

sed -E 's/(^(.....))/\1\1/g'

**3. Sa se scrie o comanda AWK care numara cate cate linii au primul cuvant egal cu ultimul.**

awk ‘BEGIN {cnt = 0}{if($1 == $NF){cnt+=1}END{print cnt}’ file

**4. Un program Shell care afiseaza numere care sunt multiplu de 15. Fara validari.**

#!/bin/bash

for arg in \*$

if [ $(($arg % 15)) -eq 0 ]; then

echo $arg

fi

done

**5. Un program Shell care afiseaza fisierele date in linia de comanda care sunt executabile si citibile.**

#!/bin/bash

for arg in\*$

if [ -x 4arg -o -w $arg -o -r $arg ]; then

echo $arg

fi

done

**6. Cate procese se creaza (fara procesul parinte) in urma executarii urmatoarelor linii de cod?**

**int i;**

**for(i = 0; i < 6; i++)**

**if(i % 3 == 1)**

**fork();**

2^2-1 = 3

**10. Enumerati 2 din 3 tipuri de alocare virtuala de memorie.**

paginata, segmentate

**11. Functie pentru crearea unui FIFO.**

mkfifo – comanda

mkfifo(const char \*pathname, mode\_t mode) – instructiune C

**12. Comportamentul functiei read(pentru pipe/FIFO) cand pipe-ul este gol fara O\_NDELAY.**

- asteapta niste date sau pana nu mai sunt scriitori, intoarce nr de biti cititi

**13. Un motiv pentru care un proces poate trece de la starea RUN in WAIT.**

- procesul intalneste o situatie unde trebuie sa astepte o resursa pentru a-si continua executia

**14. Sa se precizeze intre ce linii trebuie facuta dealocarile pentru urmatoarea secventa de cod:**

**void\* f(void\* p){**

**printf("%d\n",\*(int\*)p);**

free(p);

**return NULL;**

**}**

**int main(){**

**int i;**

**pthread\_t t[3];**

**int\* x,z;**

**z = (int\*) malloc(15\*sizeof(int));**

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

**x = (int\*) malloc(sizeof(int));**

**\*x = i;**

**pthread\_create(&t[i],NULL,f,x);**

**}**

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

**pthread\_join(t[i],NULL);**

**}**

free(z);

**return 0;**

**}**

**15. Completati programul C astfel incat suma sa se calculeze corect.**

**int sum = 0;**

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

**void \*f(void \* arg){**

pthread\_mutex\_lock(&m);

**sum++;**

pthread\_mutex\_unlock(&m);

**}**

**int main(){**

**int i;**

**pthread\_t t[10];**

**for(i = 0 ; i < 10; i++){**

**pthread\_create(&t[i],NULL,f,NULL);**

**}**

**for (i = 0 ; i < 10; i++){**

**pthread\_join(t[i],NULL);**

**}**

**return 0;**

**}**

**16. Componentele unui semafor.**

- valoare v

- coada q – contine toate procesele care asteapta

- doua operatii: P (wait), V(post)

**17. Programul A scris in programul B prin canalul "a2b", iar programul B scrie in programul A prin canalul "b2a".**

**Scrieti functiile pentru open in ordinea corecta astfel incat sa mearga programul.**

**Programul A: Programul B:**

open(‘a2b’ , O\_WRONLY ); open(‘a2b’ , O\_RDONLY );

open(‘b2a’ , O\_RDONLY ); open(‘b2a’ , O\_WRONLY );

**18. ce face exec in cazul in care se executa cu succes(nu da eroare)**

- exec reutilizeaza procesul current pentru a rula alt program. ‘Sterge’ codul procesului current si il inlocuieste cu codul noului program

**19. Ce afiseaza programul?**

**semaphore a,b,c,d;**

**a.value = 0;**

**b.value = 0;**

**c.value = 1;**

**d.value = 0;**

**while(1){**

**while(1){**

**P(a);**

**P(b);**

**}**

**print "A\n";**

**print "B\n";**

**}**

**V(d);**

**V(a);**

**P(c);**

**P(d);**

**print "C\n";**

**print "D\n";**

**V(b);**

**V(c);**

**SUB8 - 2015**

**6. How many processes are created by: fork();fork();fork();fork();** 2^4-1 = 15

**7. If the user home directory is /home/scs/abie/1234, what will the following command print?**

**cd ~/../.././xyz; pwd**

cd ~ - /home/scs/abie/1234

/../ - home/scs/abie

/../ - home/scs/

/./ - home/scs/

xyz – home/scs/xyz

pwd = home/scs/xyz

**8. What's the truth value of 0?**

- in C este considerat false

- in Shell ca si exit status este considerat true adica programul s-a executat cu success

**Sub9 – 2022**

1. **Considerand ca intr-un block incap N adrese spre alte blocuri, cate blocuri de date aunt adresate de indirectarea dubla impreuna cu cea tripla a unui 1-nod**

N^2 + N^3

1. **Dati un avantaj si un dezavantaj al alocarii cu partitii fixe absoluta in comparatie cu cea cu partitii relocabile.**

Avantaj: **Simplitate** - Fiecare proces este încărcat într-o zonă de memorie prestabilită. (pt cele relocabile: Dacă un proces este plasat într-o partiție insuficientă, acesta poate fi eliminat fără a fi executat, ceea ce poate duce la ineficiențe și probleme de performanță.)

Dezavantaj: **Ineficiența utilizării memoriei** - Fragmentare externă și internă, deoarece memoria este împărțită în partiții fixe. Spațiile de memorie pot rămâne neutilizate dacă procesele nu se potrivesc exact în ele (pt cele relocabile: Alocarea memoriei se face cu bază și deplasament, permițând plasarea proceselor în orice partiție disponibilă)

1. **Dati un avantaj si un dezavantaj al incarcarii tuturor paginilor unui program la pornirca procesului in comparatie cu incarcarea lot doar cand sunt necesare.**

Avantaj: incarcare odata, merge mai repede dupa

Dezavantaj: incarcam memoria cu data pe care e posibil sa nu le folosim

1. **Dati un avantaj si un dezavantj al politicii de plasare First-Fit fata de Worst-Fit.**

Avantaj: mai rapid

Dezavantaj: paralelecu fragmentarea

1. **Cand trece un proces din starea RUN in starea READY.**

* este intrerupt de un proces cu prioritate mai mare
* depaseste timpul de executie admis (permite altor procese sa se desfasoare)

1. **De ce un link-hard poate fi creat doar spre fisiere de pe aceeasi partitie si nu spre fisiere de pe alte partitii?**

Un link hard refera catre acelasi i-nod ca si partitia. Cum i-nodul este o structura de date utilizata pentru a reprezenta un sistem de fisiere, este intern acelui sistem. Prin urmare, nu se poate referi un i-nod din alt sistem de fisiere.

1. **Adaugati codul sursa necesar la fragmentul de cod de mai jos pentru ca print sa afiseze in consola.**

**int p[2];**

int fd = dup(1); //duplicam consola

**pipe(p);**

**dup2(p[1], 1);**

dup2(fd, 1); //scriem in consola in loc de pipe

**printf("asdf\n");**

1. **Care sunt elementele unei adrese virtuale in alocarea paginat-segmentata si ce tabele sunt implicate in calcularea adresei fizice?**

* O adresă virtuală este de forma: (s, p, d), în care s este numărul segmentului, p este numărul paginii virtuale în cadrul segmentului, iar d este deplasamentul în cadrul paginii
* In calcularea adresei fizice sunt implicate: tabela de pagini si tabele de segmente

1. **Ce va tipari fragmentul de cod de mai jos, considerand ca thread-urile se creeaza fara problema. Justificati raspunsul.**

* **﻿﻿RO: Considerati ca header -ele necesare sunt incluse aici \*/**

**void\* f(void\* a) {**

**printf("%d\n", \* (int\*)a);**

**return NULL;**

**}**

**int main() {**

* **﻿﻿RO: Considerati ca variabilele necesare sunt declarate aici \***

**for (i=0; i<10; i++) {**

**pthread\_create(&t [t], NULL, f, &);**

**}**

**/\* RO: Considerati ca aici se fac join-urile necesare \*/**

**return 0;**

1. **Care este principiul vecinatati in privinta incarcarii paginilor unui proces?**

* daca un process refera o pagina, e probabil sa refere curand paginile invecinate -> daca ni se cere o pagina aducem in memorie vecinatatea

1. **Dandu-se doua cache-uri set-asociative, unul cu 2 seturi de 4 pagini st altul cu 4 seturi de 2 pagini, care are functiona mai bine pentru secventa de cereri de pagini de mat jos? Justificati raspunsul 17, 2, 37, 6, 9**

Set 0 – pare. 2 6

Set 1 – impare 17 37 9

Set 0-

Set 1- 9 17 37(nu mai are loc)

Set 2- 6 2

Set 3-

* 2 set cu 4 pagini e mai efficient

1. **Cate thread-url ati folosi pentru a procesa un milion de fistiere? Justificati alegeres facuta.**

- In cazul oricarui program, numarul de thread-uri potrivit este necunoscut, deoarece nu putem cunoaste timpul de executie (executia programului este concurenta). Totusi, putem efectua analize asupra timpului de exectuie folosind un numar diferit de thread-uri pentru a determina o solutie optima.

1. **Cum se face calculul de adresa in alocarea cu partitii fixe absolute?**

* Calculul de adresa in alocarea cu partitii fixe absolute se face astfel : se imparte memoria in partitii, dupa care executabilul se creeaza pentru o anumita partitie, primul program ocupa prima pozitie, prima adresa ia inceputul partitiei

1. **Care este cea mai prioritara categorie de pagini de memorie din care politica de inlocuire LRU ar alege o pagina victima**

* Daca folosim acces counter: pagina cu cel mai mic counter
* Daca folosim matricea de referinta: pagina este numarul liniei cu cele mai putine cifre de 1

1. **1 avantaj si 1 dezavantaj al metodei de alocare segmentata in locul metodei de alocare paginata.**

avantaj : nu adreseaza fragmentarea

dezavantaj : grupeaza codul si datele in segmente cu protectie la acces

1. **Dandu-se un sistem de fisiere UNIX configurat cu un block de B bytes care contine A adrese si i-noduri avand S link-uri directe astfel : 1 legare simplu-indirecta, 1 legare dublu-indirecta si 1 legare triplu-indirecta. Scrieti formula pentru marimea maxima a fisierului.**

S \* B + A \* B + A^2 \* B + A^3 \* B

1. **Ce face apelul “open” inainte de deschiderea unui FIFO?**

Inainte de deschiderea unui FIFO, apelul “open” verifica existenta FIFO-ului. Daca FIFO-ul nu exista si apelul “open” este utilizat cu O\_WRONLY sau O\_RDONLY, apelul va crea FIFO-ul, blocandu-se pana cand alt proces va deschide capatul opus (citire, respectiv scriere).

1. **1 avantaj si 1 dezavantaj al alocarii cu partitii variabile in comparare cu alocarea paginate**

**Alocare cu partitii variabile:**

**Avantaj:** permite adaptarea dimensiunii partițiilor la nevoile fiecărui proces.

**Dezavantaj:** De-a lungul timpului, pe măsură ce procesele sunt create și terminate, spațiile de memorie liberă devin fragmentate. (fragmentare externa)

**Alocare paginate:**

**Avantaj:** împarte memoria în pagini de dimensiuni fixe, care sunt mapate în cadre de memorie de dimensiuni fixe. Acest lucru elimină problema fragmentării externe, deoarece orice pagină liberă poate fi utilizată pentru orice proces

**Dezavantaj:** Deoarece dimensiunile paginilor sunt fixe, este posibil ca procesele să nu utilizeze complet ultima pagină alocată, ducând la risipirea memoriei în cadrul paginilor (fragmentare internă)

1. **1 avantaj si 1 dezavantaj pentru Cache-urile set-asociative fata de cele asociative.**

Cache set-asociativ :

• avantaj : performanta (prin reducerea coliziunilor)

• dezavantaj : complexitate si costuri mai mari

Cache asociativ :

• avantaj : flexibilitate maxima in plasarea datelor (orice bloc de memorie poate fi mapat in orice linie de Cache)

• dezavantaj : lent (pagina K este pusa pe prima pozitie libera)

1. **1 avantaj si 1 dezavantaj pentru Cache-urile directe fata de cele asociative.**

Cache direct :

• avantaj : simplitate si viteza de acces

• dezavantaj : rate mari de coliziuni

Cache asociativ :

• avantaj : reducerea coliziunilor

• dezavantaj : complexitate si costuri mari