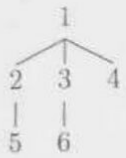
Sisteme de operare — Examen — 07.02.2019

Nume și grupă: _____ Numărul 4

1. (10p) Puteți simula o structură de directoare pe mai multe nivele cu o structură cu un singur nivel? Dacă da, explicați cum ați implementa o astfel de simulare și cum se compară cu structura pe mai multe nivele. Dacă nu, explicați ce vă împiedică să faceți acest lucru. Dați un exemplu.

2. (10p) Fie următoarea arborescență de procesese



Scrieți o secvență de cod care reproduce această structură.

3. Fie două procese P_i și P_j care vor să acceseze o zonă critică și următoarea implementare pentru P_i:

```
1  do {
2     flag[i] = true;
3     while (flag[j]) {
4        if (turn == j) {
5           flag[i] = false;
6           while (turn == j)
7              ; /* do nothing */
8           flag[i] = true;
9        }
10    }
11    /* critical section */
12    turn = j;
13    flag[i] = false;
14    /* remainder section */
15 } while (true);
```

unde procesele împart `flag[2]` (inițializat cu `false`) și `turn`.

(a) (5p) Ce se întâmplă dacă eliminăm linia 13?

(b) (10p) Arătați dacă soluția satisface cele trei proprietăți: exclusivitate mutuală, progres și timp finit de așteptare.

4. Fie două matrice $\boldsymbol{A} \in \mathbb{N}^{6\times 6}$ și $\boldsymbol{B} \in \mathbb{N}^{4\times 6}$ tinute contiguu în memorie pe linii și fie un sistem în care avem 5 *frame*-uri disponibile. În acest sistem într-o pagină încap 6 întregi, iar programele `P1` și `P2` de mai de jos încap fiecare separat într-o pagină.

```
1 P1:
2 for (i = 0; i < 6; i++)
3    for (j = 0; j < 6; j++)
4       A[i][j] = i+2*j;
```

```
1 P2:
2 for (i = 0; i < 4; i++)
3    for (j = 0; j < 6; j++)
4       B[i][j] = i+j;
```

Presupunem că programele se execută concurent astfel: fiecare program stă pe procesor cât să ducă până la capăt instrucțiunea de la linia 4 o singură dată după care cedează locul concurentului.

(a) (5p) Cum arată programele și datele repartizate pe pagini?

(b) (5p) Cum arată diagrama Gannt folosind algoritmul LRU cu o strategie de înlocuire a paginilor globală (în care ambele programele pot folosi toate *frame*-urile disponibile în sistem).

(c) (5p) Cum ați aloca *frame*-urile pentru o strategie de înlocuire a paginilor locală (în care fiecare program primește un număr fix de *frame*-uri pe care doar el le poate utiliza). De ce?

Sisteme de operare — Examen — Numărul 2

Nume și grupă: [redacted]

1. Fie următoarea secvență

```
for (i=0; i<2; i++)
  for (j=i; j>0; j--)
    fork();
```

   (a) (5p) Câte procese sunt create?

   (b) (5p) Desenați arborescența proceselor create.

2. (10p) Considerați problema filosofilor și soluția propusă mai jos pentru $n \in \mathbb{N}$ filosofi așezați la masă.

```
do {
  wait(chopstick[i]);
  wait(chopstick[(i+1)%n]);
  /* ... */
  signal(chopstick[i]);
  signal(chopstick[(i+1)%n]);
} while (true);
```

   (a) (5p) Demonstrați că fenomenul de *deadlock* apare pentru $n = 5$ filosofi.

   (b) (5p) Dacă modificăm soluția astfel încât fiecare filosof poate ridica betele doar dacă ambele sunt disponibile, fenomenul *deadlock* dispare dar apare fenomenul de *starvation*. Arătați de ce.

3. Fie următoarea secvență de procese care apar la diferite momente de timp t

Proces	t	CPU
P_0	0	5
P_1	0	3
P_2	2	8
P_3	4	1
P_4	6	7

   (a) (5p) Cum arată diagrama Gantt rezultată în urma aplicării algoritmului Round Robin *nonpreemptive*?

   (b) (5p) Dar pentru același algoritm în modul *preemptive* cu o cuantă de timp $q = 2$?

4. (10p) Care este numărul minim de *frame*-uri necesar pentru execuția corectă a proceselor pe un procesor cu instrucțiuni de tipul: `instr reg, memop, memop` (ex. `add r8, [0xdead], [0xbeef]`).

5. Fie un disk cu 5000 de cilindri și următoarea coadă de cereri *I/O* în așteptare 2000, 3000, 1200, 4, 2018. Fiecare intrare reprezintă un cilidru, iar capul de citire al disk-ului se află la poziția 1000 și a fost înainte la poziția 314.

   (a) (5p) Începând de la poziția curentă, care este ordinea și distanța totală parcursă de cap pentru a satisface toate cererile din coadă folosind algoritmul FCFS?

   (b) (5p) Dar folosind algoritmul SCAN?

Sisteme de operare

Examen

Numărul 2

Nume și grupă: [redacted]

1. (a) (5p) Ce este un proces *orfan*?

   (b) (5p) Scrieți o secvență scurtă de cod și arătați când un proces devine *orfan*.

2. Fie următoarea secvență

```
for (i=0; i<3; i++) {
  pthread_create();
  fork();
  fork();
}
```

   (a) (5p) Câte procese și fire de execuție sunt create? La numărare, ce presupunere ați făcut legată de `fork()`?

   (b) (5p) Desenați arborescența proceselor și firelor de execuție create. Etichetați cu P procesele și cu T firele de execuție.

3. Considerați problema filosofilor și soluția propusă mai jos pentru $n \in N$ filosofi așezați la masă.

```
do {
  wait(chopstick[i]);
  wait(chopstick[(i+1)%n]);
  /* ... */
  signal(chopstick[i]);
  signal(chopstick[(i+1)%n]);
} while (true);
```

   Această soluție permite apariția fenomenului de *deadlock*.

   (a) (5p) Modificați soluția ridicând asimetric bețișoarele: filosofii impari ridică întâi bețișorul din dreapta, cei pari pe cel din stânga. Arătați că nu mai apare fenomenul.

   (b) (10p) Arătați dacă noua soluție satisface cele trei proprietăți: exclusivitate mutuală, progres și timp finit de așteptare.

4. Fie o matrice $A \in N^{10\times10}$ ținută contiguu în memorie pe linii și fie un sistem în care avem 3 *frame*-uri disponibile. În acest sistem într-o pagină încap 10 întregi, iar programele P1 și P2 de mai de jos încap în totalitate într-o pagină.

P1:
```
for (i = 0; i < 10; i++)
  for (j = 0; j < 10; j++)
    A[i][j] = 0;
```

P2:
```
for (j = 0; j < 10; j++)
  for (i = 0; i < 10; i++)
    A[i][j] = 0;
```

   (a) (5p) Cum arată programul și datele repartizate pe pagini?

   (b) (5p) Folosind algoritmul LRU, care este programul eficient? De ce?

   (c) (5p) Cum arată diagramele Gannt pentru P1 și P2?