Gmail

"Sisteme de operare" - testul 2 de laborator

Google Forms <forms-receipts-noreply@google.com> 6 June 2022 at 10:03

Google Forms

Thanks for filling in "Sisteme de operare" - testul 2 de laborator

Here's what was received.

"Sisteme de operare" - testul 2 de laborator

Testul online de evaluare a ultimelor 6 săptămâni din materia practică la SO.

CITIȚI CU MARE ATENȚIE FIECARE ÎNTREBARE! În dreptul fiecărei întrebări este menționat punctajul ce va fi atribuit DOAR pentru răspunsul CORECT la acea întrebare.

Email *

Datele personale, pentru autentificare.

Introduceți-vă numele complet (cu litere MAJUSCULE), în formatul: NUME Inițiala-tatălui PRENUME *

Selectați grupa din care faceți parte în catalog (Nu cea la care ați participat la laboratoare!). *

Anul 1, grupa B2

Numărul dumneavoastră matricol, specificat cu cifre și litere MAJUSCULE, fără spații albe: *

Exerciții aplicative

Al treilea formular cu întrebări de evaluare.

Se consideră programul din figura alăturată. Bifați TOATE afirmațiile INCORECTE de mai jos. Observații: 1. Se consideră incluse toate declarațiile #include necesare. 2. Se presupune că secțiunile de cod A, B și C nu conțin apeluri exec sau exit (și nici return). (1p)

```
int main()
{
    /* Secventa de cod A */
    pid_t pid1;
    if(-1 == (pid1=fork()) )
    {
        perror("Eroare la fork"); return 1;
    }
    if(pid1 != 0)
    {
        /* Secvență de cod B */
    }
    else
    {
        /* Secvență de cod C */
    }
    /* Secvență de cod D */
    return 0;
}
```

- ✓ Procesul tată va executa doar secvențele de cod A și D.
- ✓ Procesul fiu nu va executa secvența de cod B.
- ✓ Procesul fiu va executa doar secvențele de cod B și C.
- ✓ Procesul tată va executa doar secvența de cod A.
- ✓ Procesul tată va executa secvența de cod B.
- ✓ Procesul fiu nu va executa secvența de cod D.

Se execută următorul program, în care presupunem că apelul fork() nu eșuează, iar programul „cmd” returnează valoarea 0 numai dacă se execută cu succes. Bifați TOATE afirmațiile adevărate de mai jos. Observație: se consideră că programul are incluse toate declarațiile #include necesare. (1p)

```
int main(int argc, char *argv[])
{
  int status;
  switch( fork() )
  {
    case -1: perror("Eroare la fork"); return 1;
    case  0: execlp("cmd","cmd",...,NULL); printf("1"); break;
    defaut : wait(&status);
             if( WIFEXITED(status) ) {
               switch( WEXITSTATUS(status) ) {
                 case 0 : printf("2"); break;
                 case 1 : printf("3"); break;
                 default: printf("4");
               }
             }
             else printf("5");
             return 0;
  }
  return 2;
}
```

- [x] Dacă apelul funcției execlp eșuează, atunci pe ecran se va afișa (și) cifra 1.
- [x] Dacă apelul funcției execlp eșuează, atunci pe ecran se va afișa (și) cifra 4.
- [ ] Tatăl afișează (și) cifra 2, indiferent de succesul sau eșecul execuției apelului execlp.
- [ ] În cazul în care programul "cmd" și-a terminat execuția cu succes în procesul fiu, atunci tatăl afișează (și) cifra 3.
- [ ] În cazul în care programul "cmd" și-a terminat execuția cu insucces în procesul fiu, atunci tatăl afișează 14.
- [x] În cazul în care programul "cmd" a fost terminat forțat în procesul fiu, atunci pe ecran se va afișa (și) cifra 5.

Se consideră următorul program C. Pentru procesul părinte și, respectiv, pentru procesele fii cu PID-urile pid1 și respectiv pid2, bifați TOATE secțiunile de cod care vor fi EFECTIV executate de procesul respectiv. Observații: 1. Se consideră că programul conține toate declarațiile #include necesare. 2. Se consideră că nu există apeluri exec, exit sau return în secțiunile de cod A-E. (3 întrebări x 1p fiecare)

```
int main()
{
  /* Sectiunea de cod A */
  pid_t pid1, pid2;
  pid1 = fork();
  if (pid1 != -1)
  {
    if (pid1 == 0)
    {
       /* Sectiunea de cod B */
       if (-1 == (pid2 = fork()) { perror("Eroare: "); exit(1); }
       if (pid2 != 0)
          {
             /* Sectiunea de cod C */
          }
       else
          {
             /* Sectiunea de cod D
          }
    }
   /* Sectiunea de cod E */
  }
  else
     { perror("Eroare: "); exit(2); }

  return 0;
}
```

	A	B	C	D	E
procesul părinte (inițial)	✓	☐	☐	☐	✓
fiul cu PID-ul pid1	☐	✓	✓	☐	✓
fiul cu PID-ul pid2	☐	☐	☐	✓	✓

Se consideră următorul program, prin care se dorește implementarea unui comportament similar comenzii compuse: A && B | C . Presupunem că niciuna dintre funcțiile apelate nu dă eroare. Bifați corespondențele dintre comenzile cmd1, cmd2, respectiv cmd3, și afirmațiile de pe coloane. Observație: se consideră incluse toate declarațiile #include necesare. (3 întrebări x 1p fiecare)

```
int main(int argc, char *argv[])
{
  int status,p[2];
  switch( fork() ) {
    case 0: execlp("cmd1","cmd1",...,NULL);
    defaut: wait(&status);
            if(! WIFEXITED(status)) return 10;
            switch( WEXITSTATUS(status) ) {
              case 0 : pipe(p);
                       switch( fork() ) {
                         case 0 : close(p[1]); dup2(p[0],0);
                                  execlp("cmd2","cmd2",...,NULL);
                         default: dup2(p[1],1);
                                  execlp("cmd3","cmd3",...,NULL);
                       }
              default: return 20;
            }
  }
  return 0;
}
```

	se execută în cadrul procesului părinte inițial	se execută în cadrul primului proces fiu creat	se execută în cadrul celui de-al doilea proces fiu	această comandă va fi A-ul din A&&B|C	această comandă va fi B-ul din A&&B|C	această comandă va fi C-ul din A&&B|C
comanda cmd1	☐	☑	☐	☑	☐	☐
comanda cmd2	☑	☐	☐	☐	☐	☑
comanda cmd3	☐	☐	☑	☐	☑	☐

Se consideră programul din figura alăturată. Specificați ce mesaj se va afișa pe ecran în urma execuției sale. Bifați și opțiunea Other/Altele, în care să scrieți justificarea alegerii făcute. Observație: se consideră incluse toate declarațiile #include necesare. (2p)

```
int main()
{
   int fdp, fdc, pid, nr;

   if (-1 == mkfifo("fifo_EF1A30",0600) ) { perror("mkfifo"); exit(1); }
   fdp = open("fifo_EF1A30",O_WRONLY);
   if (-1 == fdp) { perror("open1"); exit(2); }

   if (-1 == (pid = fork()) ) { perror("fork"); exit(3); }
   if (pid == 0) {
     char buffer[100];
     fdc = open("fifo_EF1A30",O_RDONLY);
     if (-1 == fdc) { perror("open2"); exit(4); }
     if ( 14 == read(fdc,buffer,13+1) )
         printf("Citire reusita integral\n");
     else
         printf("Probleme la citire\n");
     return 0;
   }

   char* text="Salut, fiule!";
   nr = write(fdp,text,strlen(text)+1);
   if (-1 == nr) { printf("Scriere esuata\n"); exit(5); }
   wait(NULL);
}
```

- ☐ open 1 : file not found
- ☐ open 2 : file not found
- ☐ Scriere esuata
- ☑ Citire reusita integral
- ☐ Probleme la citire
- ☐ Nu se afișează nimic
- ☑ Other: Numele canalului este acelasi iar flagurile de citire/scriere sunt ok, deci mesajul e transmis corect

[Quoted text hidden]