Thread-uri

1. Diferenta dintre procese si thread-uri

**Procese:**

* un proces este un calcul care poate fi executat concurent cu alte calcule.
* abstractizare a activitatii procesului
* un proces este conditionat de 3 factori: procedura, procesor, mediu
* un proces este reprezentat in memorie prin urmatoarele zone: zona de cod, zona datelor globale, zona heap si zona stiva

**Threaduri:**

* un thread este o entitate de executie din interiorul unui proces, compusa dintr-un context si o secventa de instructiuni de executat
* procese concurente
* incapsularea instructiunilor in functia threadului
* tratare independenta
* ocupa doar spatiul de stiva
* pot fi executate in progres sau paralel

1. Functii importante pentru thread-uri

**Crearea unui thread**

pthread\_create(pthread\_t\* id, pthread\_attr\_t\* attr, void\* (\*func)(void\*), void\* arg);

id - id-ul thread-ului

atribute pentru creare

functia asociata thread-ului

arg - argumentul cu care este apelata fumctia

Returneaza 0 daca thread-ul a fost crea cu succes si un cod de eroare altfel ( id - nu este setat si va avea o valoare nedefinita )

**Iesirea dintr-un thread**

pthread\_exit(int\* status)

se opreste executia thread-ului curent

prin variabila status se poate returna o valoarea

**Asteptarea unui thread**

pthread\_join( pthread\_t\* id, void\*\* status)

id - id-ul thread-ul

status - statusul care thread-ul l-a returnat

Se asteapta terminarea executiei thread-ului cu id-ul dat

Daca nu exista thread-ul cu id-ul dat se asteapta terminarea oricarui thread

1. Sincronizari

Mutex-ul reprezinta de asemenea o variabila globala, care este blocata in momentul in care un thread foloseste o resursa critica, pentru ca aceasta sa nu poata fi accesata de catre alt thread pana cand nu se face unlock pe mutex.

**Initializare statica**

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

**Initializare dinamica**

pthread\_mutex\_init(pthread\_mutex\_t\* mutex, pthread\_mutexattr\_t\* mutexattr)

**Blocarea variabilei mutex:**

pthread\_mutex\_lock(pthread\_mutex\_t\* mutex)

**Deblocarea variabilei mutex:**

pthread\_mutex\_unlock(pthread\_mutex\_t\* mutex)