Facultatea: **Automatică şi Calculatoare**
Domeniul: **Calculatoare şi Tehnologia Informației**

An universitar: **2016 – 2017**

Sisteme de Operare

- Gestiunea proceselor

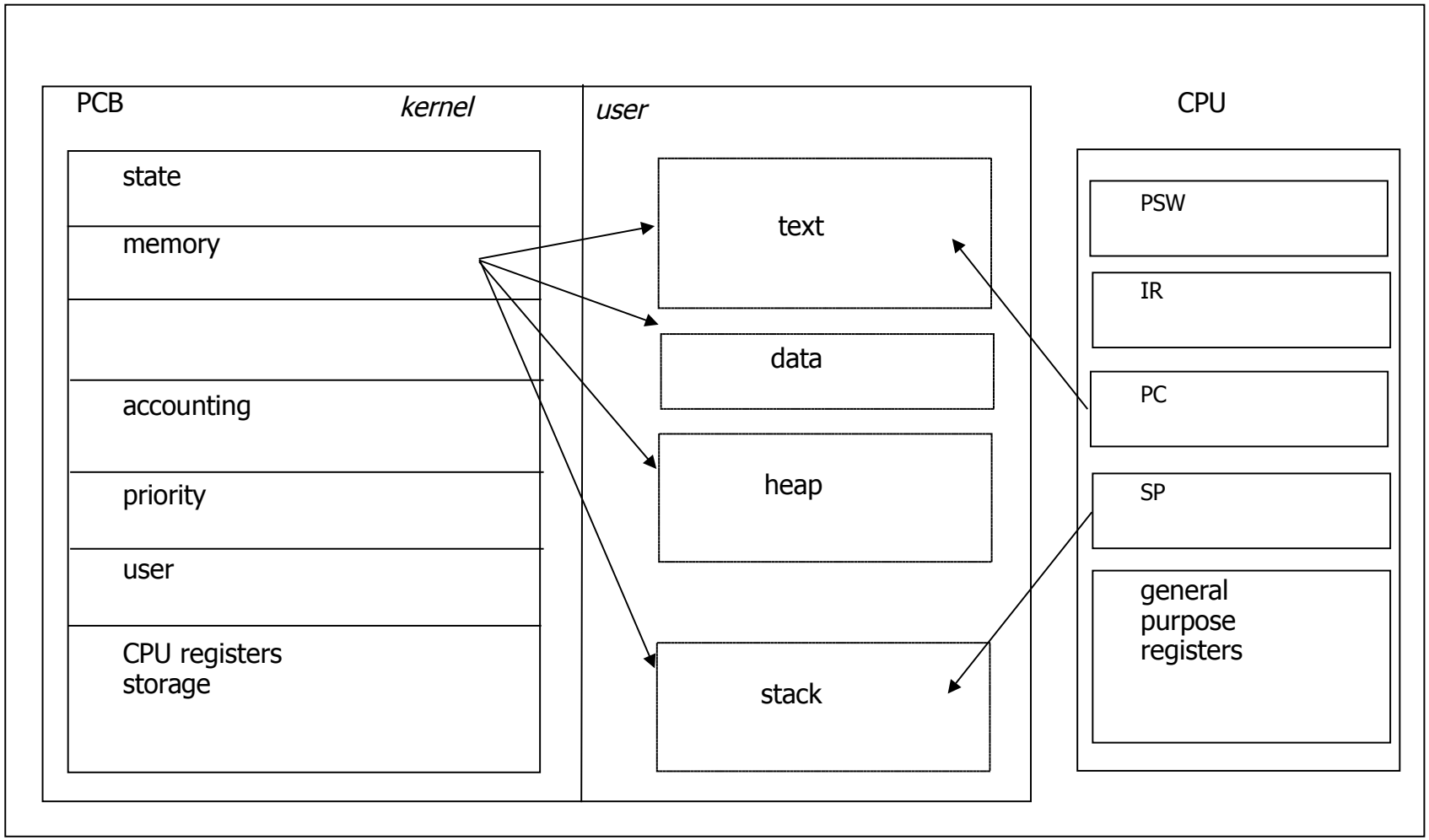
Priorităţile unui proces

- Priorităţile sunt dinamice şi se calculează din secundă în secundă
- Exemplu UNIX:
  - **prioritate = baza + utilizare_recentă_CPU / constantă + valoare_nice**
    - **baza** = 60 (ceasul întrerupe de 60 ori/sec);
    - **valoare_nice** se presupune nulă;
    - nucleul calculează **utilizare_recentă_CPU** prin împărţirea numărului de tacţi la 2 în momentul calculului priorităţii;
    - **constanta** este 2.

Process Control Block (PCB)

- Zonă ce face parte din imaginea unui proces pe lângă zonele text, date şi stivă
- Furnizează date cu privire la:
  - identificarea procesului: pid, uid, real uid, real gid, effective uid, efective gid
  - informaţiile de stare: regiştrii vizibili utilizatorului, de obicei de la 8 la 32, uneori pe unele maşini RISC peste 100
  - informaţiile de control al procesului

Process Control Block (PCB)



Structuri de date pentru gestiunea proceselor (Unix)

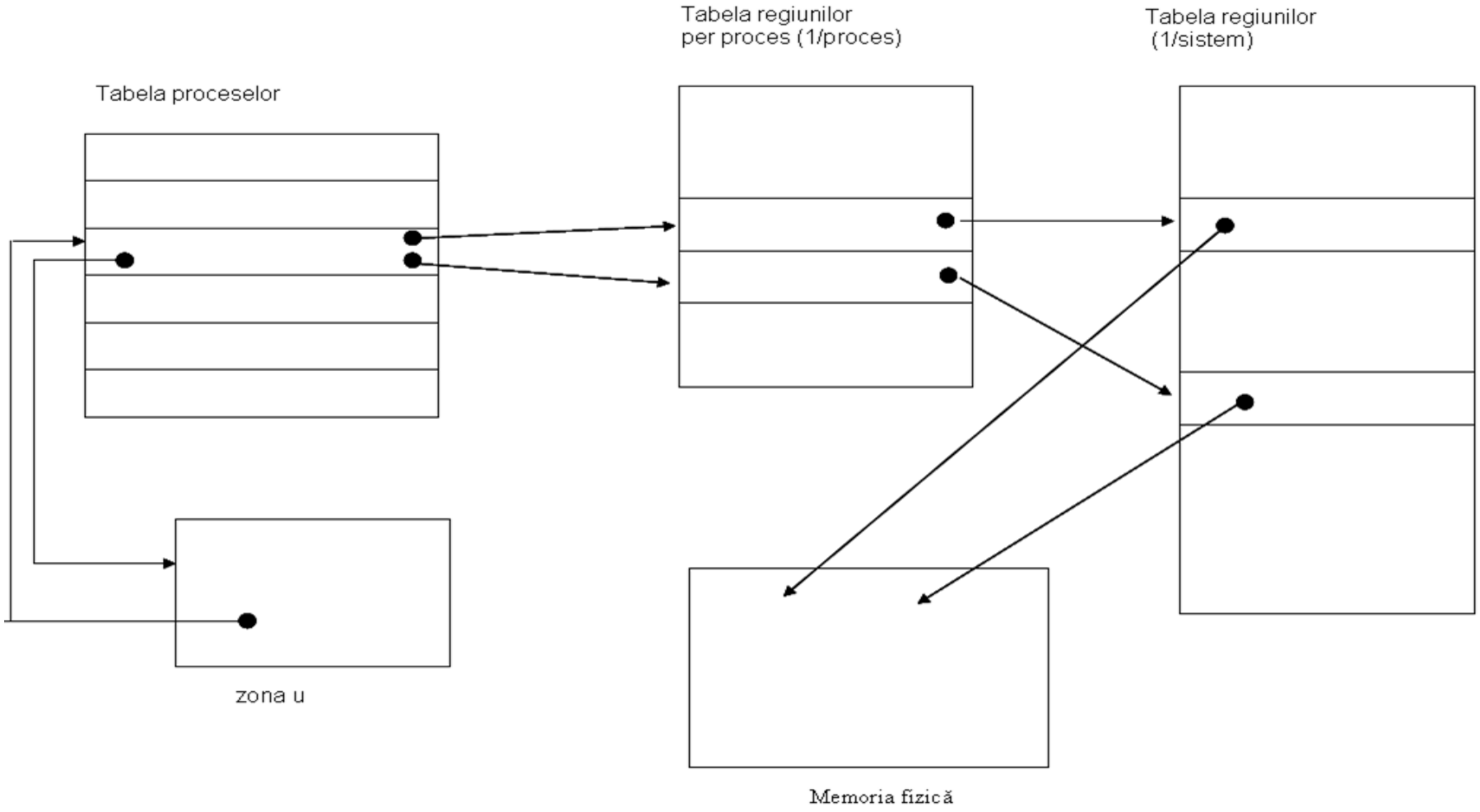


Tabela proceselor

- unică
- alocată in mod static de nucleu
- dimensiunea este fixată la generarea sistemului
- limitează numărul de procese pe care nucleul le poate gestiona pe sistemul respectiv.
- câmpuri:
  - starea procesului;
  - localizarea procesului în memoria internă sau în memoria secundară folosită pentru swapping;
  - dimensiunea procesului;
  - identificatorii ataşaţi utilizatorului şi grupului său;
  - identificatorul procesului;
  - descriptorul evenimentului care a produs trecerea procesului în starea de aşteptare;
  - parametrii de planificare pentru obţinerea procesorului;
  - semnalele trimise procesului, dar încă netratate;
  - diferiţi timpi care indică timpul de execuţie în mod utilizator şi în mod nucleu pentru calculul priorităţii procesului;
  - un câmp este folosit pentru SIGALARM.

Zona u (U area)

- este generată şi ataşată unui proces la crearea lui
- este accesibilă nucleului numai în timpul execuţiei procesului la care este ataşată
- Variabila **u** folosită de nucleu conţine adresa virtuală a **zonei u** a procesului în curs de execuţie
- Câmpurile zonei conţin:
  - un pointer la intrarea în tabela proceselor corespunzătoare procesului la care este ataşată zona u;
  - identificatorul utilizatorului real şi efectiv, în funcţie de care se stabilesc drepturile de acces la fişiere, cozile de mesaje, memorie comună, semafoare, etc.
  - timpii de execuţie ai procesului şi descendenţilor săi în mod utilizator şi nucleu;
  - modul de reacţie a procesului la semnale (ignorate, prelucrate, tratare de nucleu);
  - identificatorul terminalului de control („login terminal”) asociat cu procesul;
  - eroarea apărută în timpul apelului unei funcţii de sistem;
  - valoarea returnată de o funcţie de sistem;
  - parametrii de I/O: tipul transferului, adresa din spaţiul procesului unde/de unde se transferă, deplasamentul în fişier etc;
  - directorul curent;
  - tabela descriptorilor de fişiere utilizator (TDFU);
  - dimensiunea limită a procesului şi a fişierelor;
  - masca pentru drepturile de acces la fişierele create

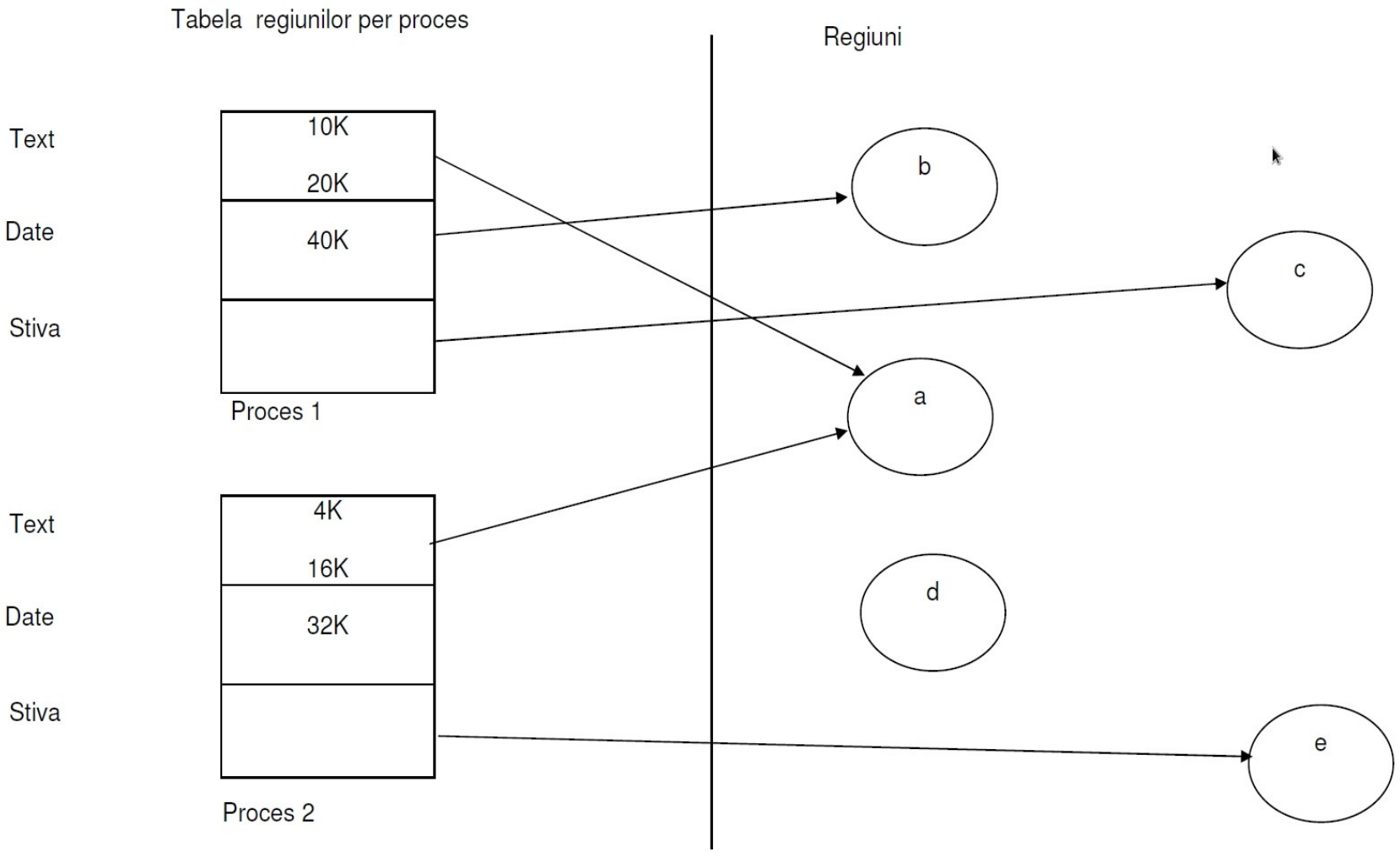
Tabela regiunilor

- este gestionată de nucleu
- O intrare în tabelă conţine informaţiile necesare identificării unei regiuni în memoria fizică internă
- O regiune este o zonă contiguă din spaţiul de adrese virtual, care este tratat ca un obiect distinct ce poate fi partajat sau protejat
- Câmpurile:
  - un pointer la i-node-ul fişierului al cărui conţinut se găseşte în regiune;
  - tipul regiunii: text, date, memorie partajată, stivă;
  - dimensiunea regiunii;
  - adresa regiunii în memoria internă;
  - starea regiunii (poate fi o combinaţie de: blocat, în cerere, în curs de încărcare, validă – conţinutul este încărcat în memorie şi accesibil);
  - numărul de procese care referenţiază regiunea.

Tabela regiunilor per proces

- asociată unui proces
- poate intra în **tabela proceselor**, în **zona u**, sau într-o **zonă de memorie alocată acestui scop**
- cele mai multe implementări plasează această zonă în tabela proceselor
- conţine:
  - un pointer la intrarea corespunzătoare în tabela regiunilor;
  - adresa virtuală de început (start) a regiuni; pentru procese diferite, o regiune partajată poate vedea adrese virtuale de început diferite;
  - drepturile de acces la regiune: read-only, read-write, read-execute.
  - conceptul de regiune este independent de tehnica de gestionare a memoriei prin paginare la cerere, prin partiţionare dinamică şi swapping, prin segmentare etc.

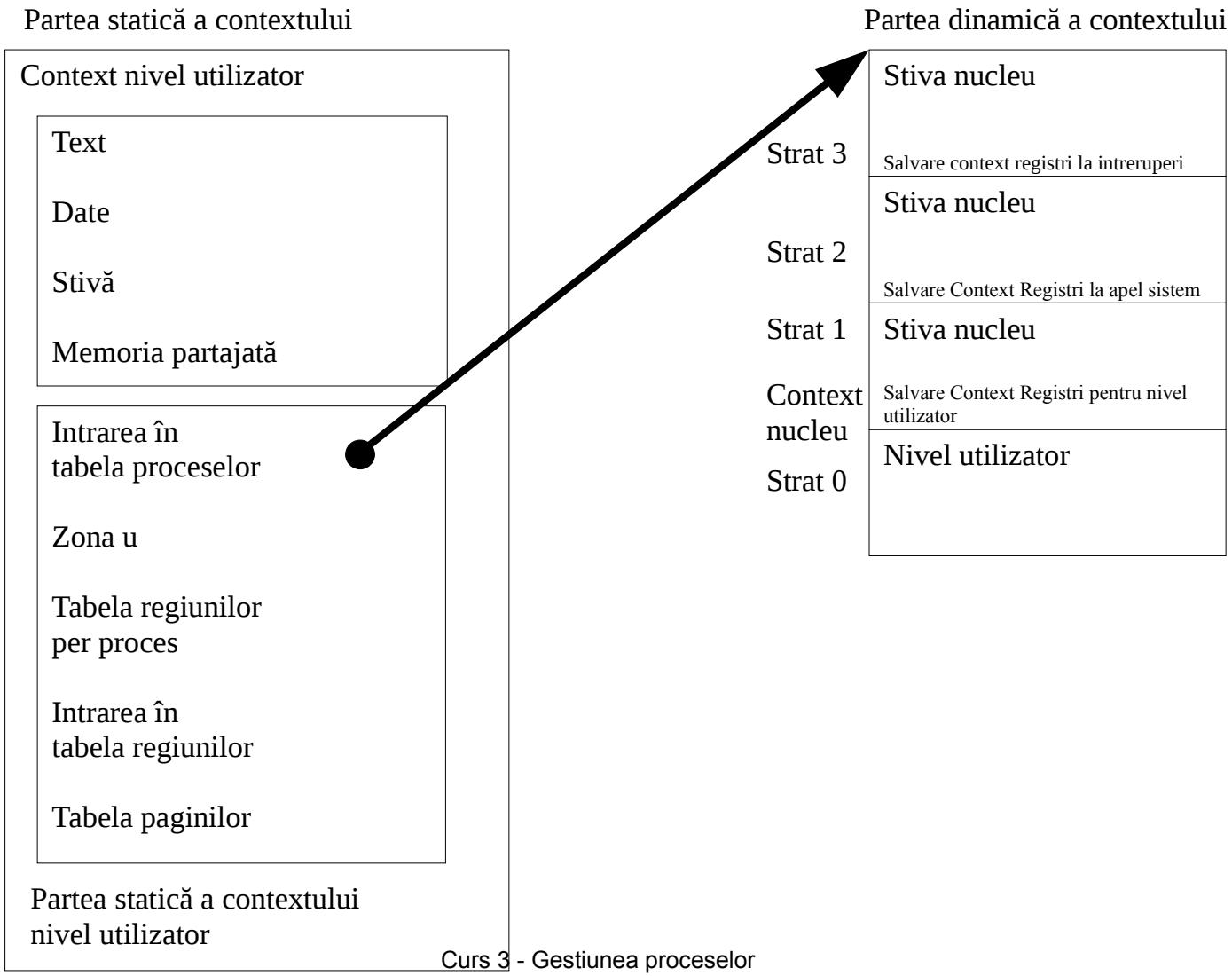
Partajarea unei regiuni



Contextul unui proces

- **Preemptarea** – reprezintă acţiunea prin care se întrerupe execuţia unui task, fără cooperarea acestuia, cu intenţia de a se putea continua execuţia din acelaşi punct la o data ulterioară.
- Pentru realizarea acestui lucru este necesară realizarea unei operaţii denumită **schimbare de context** (**context switch**).
- Această acţiune este realizată de către un task privilegiat, sau de o parte a sistemului de operare numită **planificator**, ce are posibilitatea de a preempta sau întrerupe şi a relua execuţia altor task-uri din sistem.
- **Contextul unui proces** reprezintă un set minimal de date utilizate de un proces, date ce trebuie salvate pentru a se putea permite întreruperea execuţiei unui proces într-un anumit moment şi continuarea execuţiei la o data ulterioară.

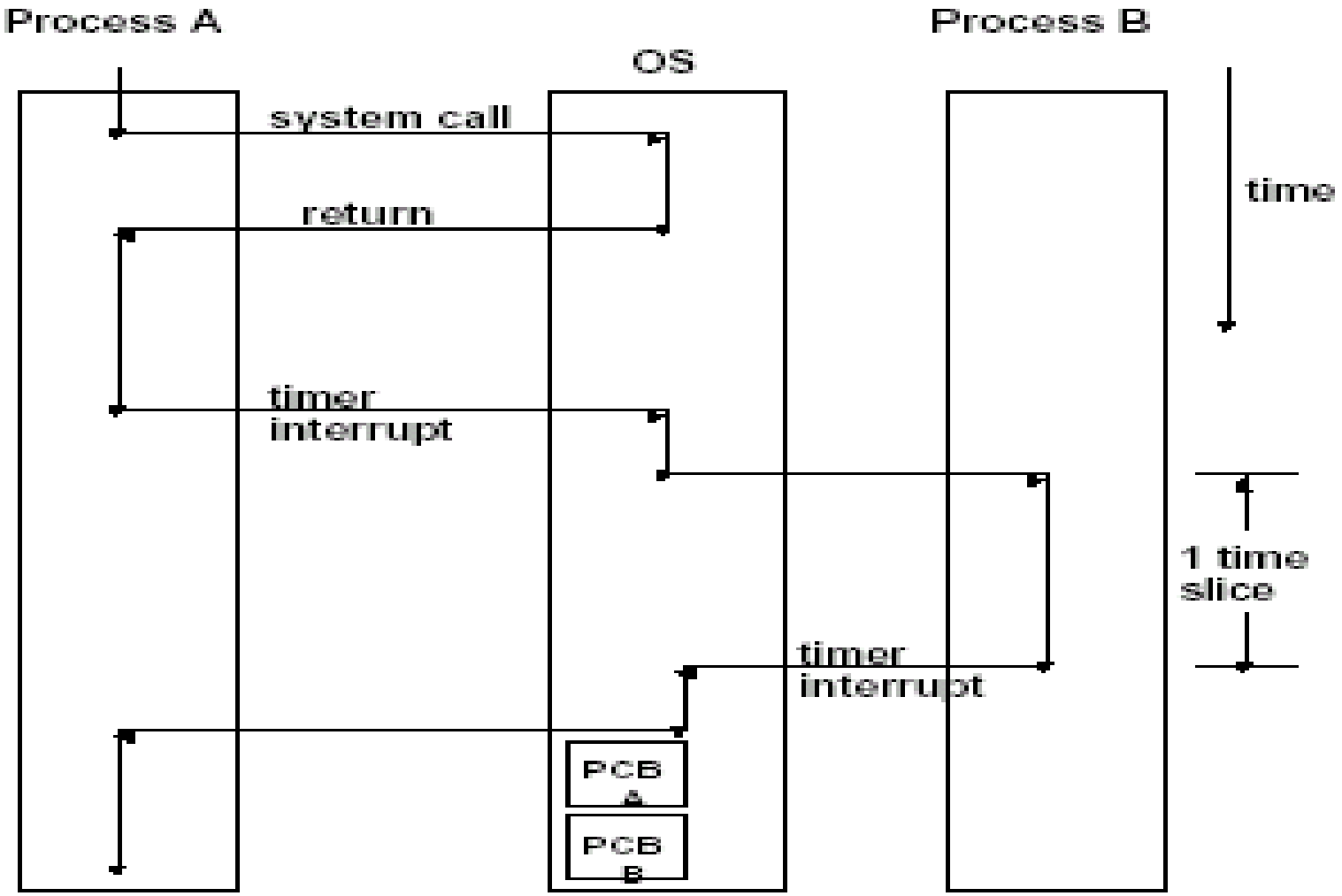
Contextul unui proces (2)



Contextul unui proces (3)

- **contextul utilizator** (spaţiul de adrese utilizator):
  - segmentul de text
  - segmentul de date (inclusiv cele partajate din memoria comună)
  - segmentul stivă-utilizator;
- **contextul registrelor** (registrele hardware):
  - numărătorul de instrucţiuni
  - registrul de stare program
  - registrul de stivă (“stack pointer” – stiva referită (utilizator sau nucleu) este funcţie de modul de execuţie: modul utilizator sau modul nucleu),
  - registrii generali.
- **contextul nivel nucleu** (structurile de date ale nucleului legate de proces):
  - intrarea în tabela proceselor
  - zona u
  - tabela regiunilor per proces, intrarile corespunzătoare în tabela regiunilor (în cazul gestionării memoriei prin paginare la cerere). Regiunile partajate de text sau date se consideră că fac parte din contextul fiecărui proces care le partajează.
  - stiva nucleu – conţine părţile stivei referitoare la apelul funcţiilor nucleului, care se execută în modul nucleu. Cind procesul se execută în mod utilizator stiva nucleu este goală.

Schimbarea de context

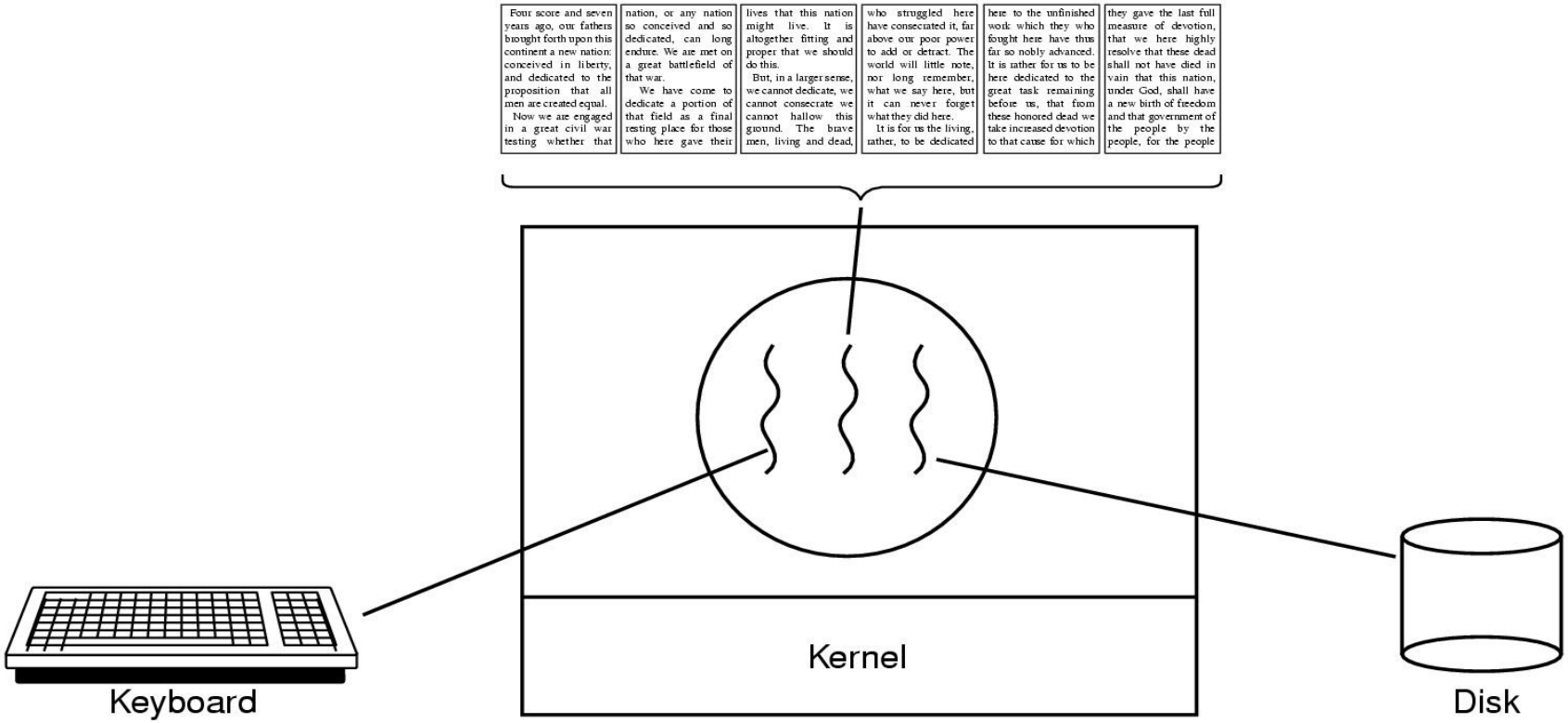


Schimbarea de context (2)

- Etapele schimbării de context:
  - decizia dacă trebuie să se execute o "schimbare de context";
  - salvarea contextului vechiului proces;
  - alegerea noului proces de către planificatorul de procese;
  - restaurarea contextului noului proces.

Thread-uri (fire de execuție)

- Definiţie:
  - Un thread este o entitate distinctă în cadrul unui proces (cea mai mică unitate de procesare), pe care kernelul o poate planifica pentru execuţie şi reprezintă unul sau mai multe subtask-uri în cadrul task-ului executat de un proces



Fire de execuție într-un procesor de texte

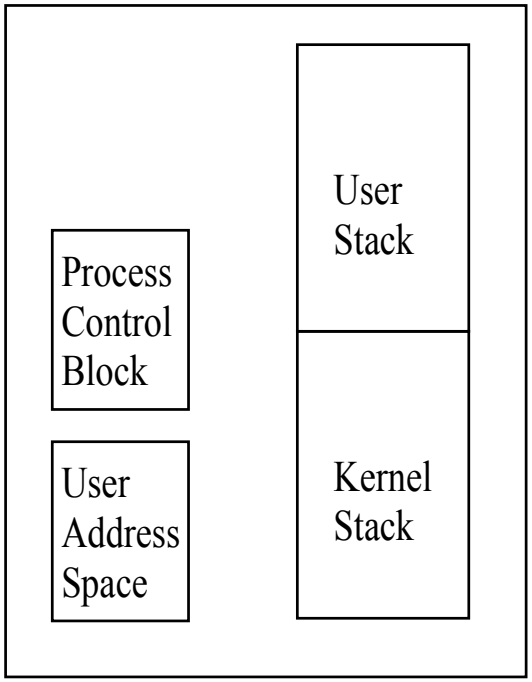
Thread-uri (2)

Carateristici:

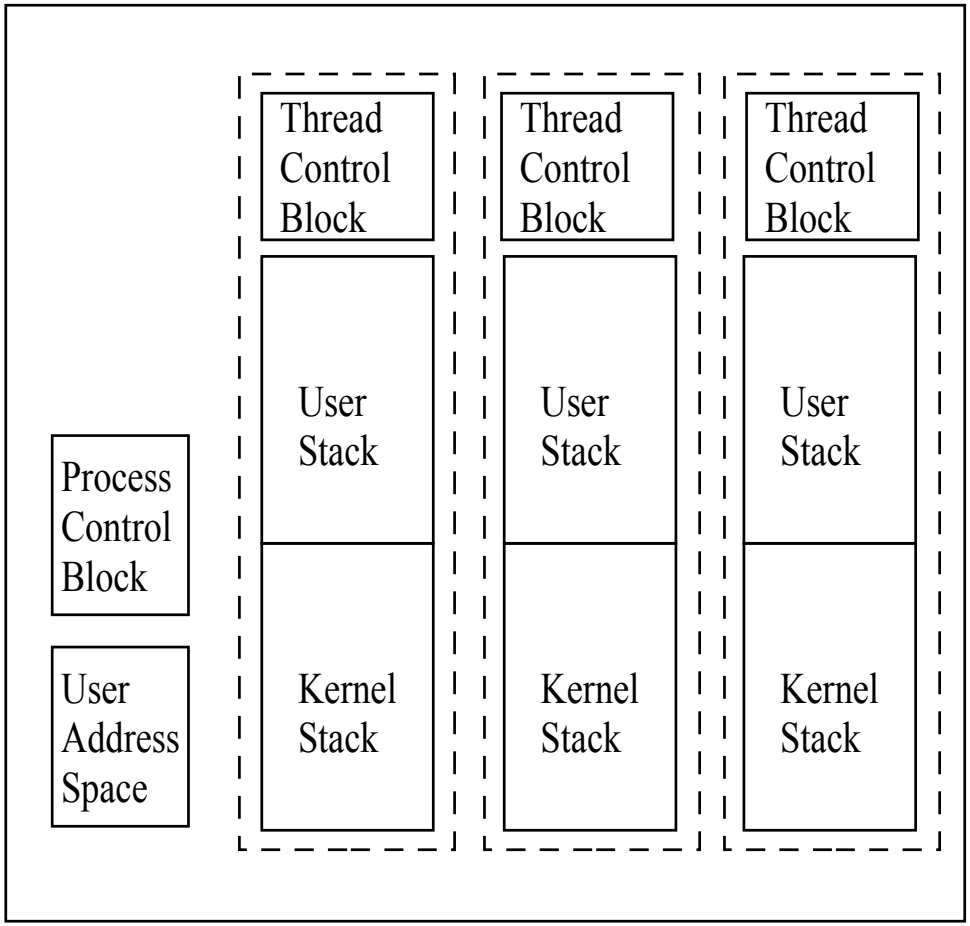
- are propria sa stare de execuţie (running, ready etc),
- are stivă proprie
- are context propriu
- are acces la memoria şi resursele procesului (partajate între firele de execuţie).
- îi poate fi întreruptă execuţia.

Modelul procesului cu unul sau mai multe thread-uri

Modelul procesului cu un singur thread



Modelul procesului cu mai multe thread-uri



Relația între thread-uri şi procese

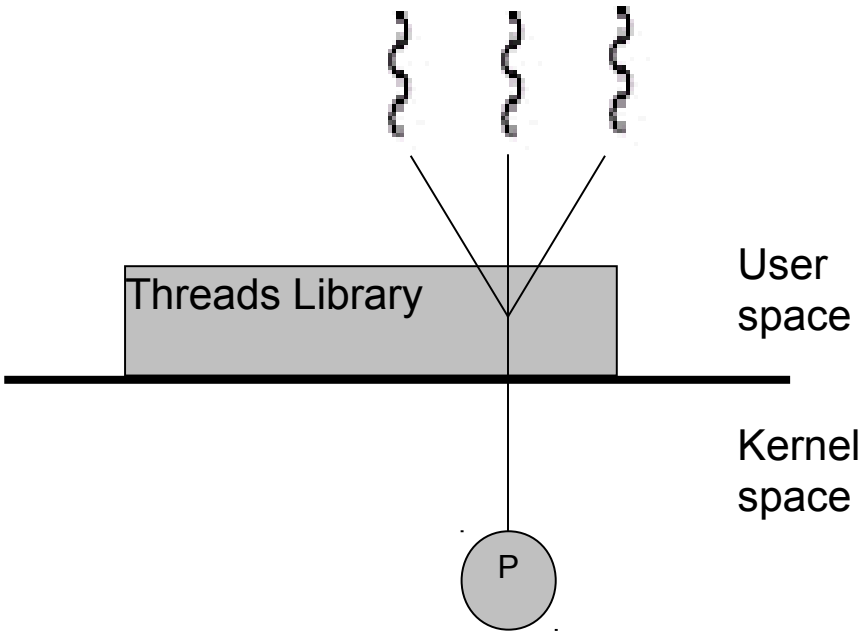
- 1 la 1:
  - fiecare proces are un singur thread (sistemele Unix – la începutul lor) ;
- n la 1:
  - fiecare proces poate avea mai multe thread-uri (cel mai folosit caz – Win2000, Linux, Solaris, OS/2, Match);
- 1 la n:
  - întâlnit în unele sisteme distribuite; un thread se poate muta pe maşini diferite şi în spaţii diferite (Ra (Clouds), Emerald);
- n la n:
  - combinaţie între cele două de mai sus: poate comuta între domenii diferite pentru a realiza anumite operaţii cheie

Thread-uri

- Operaţiile care se pot realiza asupra thread-urilor
  - spawn: - crearea unui nou thread;
  - block: - aşteptarea unui eveniment;
  - unblock: - apariţia evenimentului duce la deblocare;
  - finish: - terminarea thread-ului.

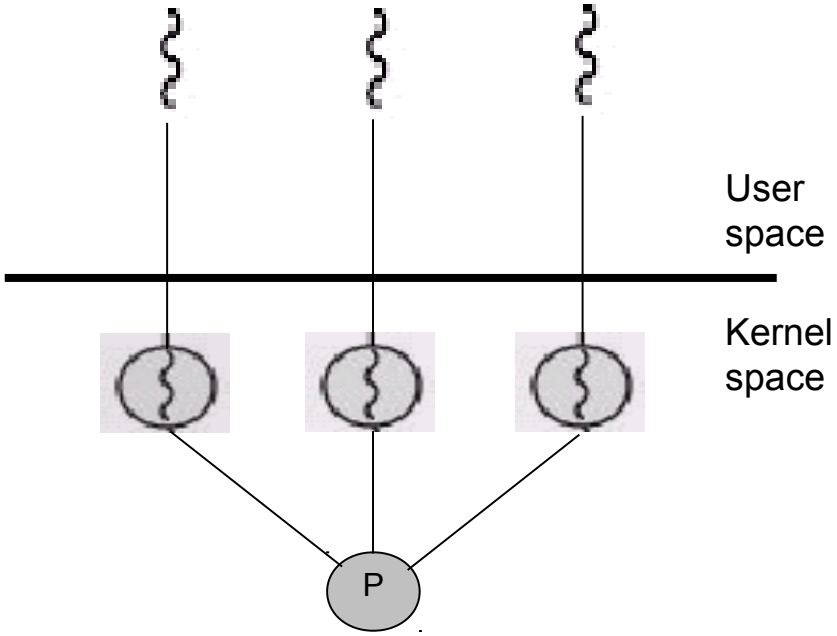
Implementarea şi gestionarea thread-urilor

- la nivel utilizator:
  - aplicaţiile creează şi gestionează thread-urile prin anumite funcţii de bibliotecă;
  - sistemul gestionează procesul ca un întreg;
  - pot apare probleme la apelurile sistem blocante – când un thread se blochează sunt blocate toate thread-urile procesului;
  - nu suportă sisteme multiprocesor, pot rula pe orice sistem de operare.



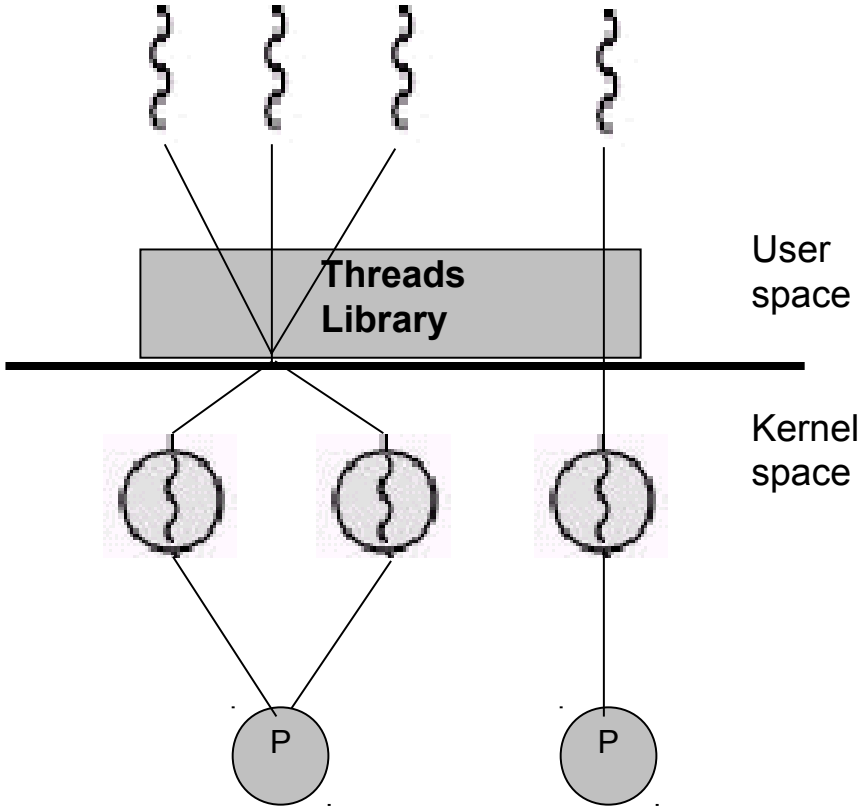
Implementarea şi gestionarea thread-urilor (2)

- la nivel kernel:
  - kernelul gestionează thread-ul;
  - suport pentru sistemele multiprocesor;
  - kernelul însuşi poate avea mai multe thread-uri;
  - este nevoie de comutarea între modurile user şi kernel
  - Exemplu: Windows 2000 şi OS/2.



Implementarea şi gestionarea thread-urilor (3)

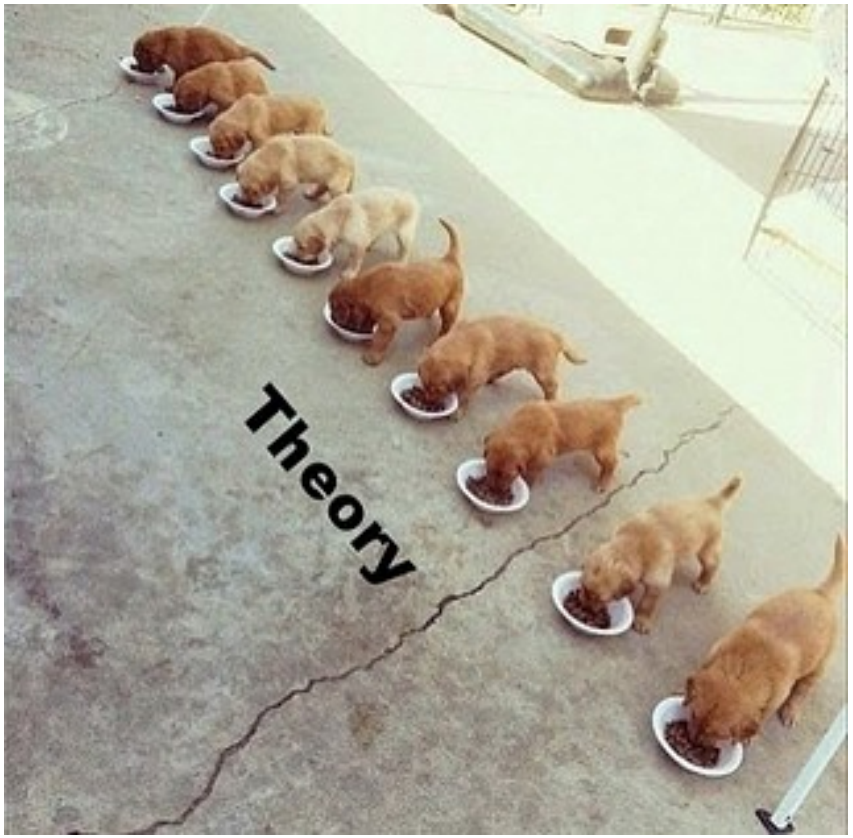
- combinaţie între nivel utilizator şi kernel:
  - fiecare thread al kernelului poate avea mai multe thread-uri utilizator;
  - crearea thread-urilor se face în spaţiul utilizator (exemplu: Solaris).



Thread-uri

- Avantaje:
  - un singur proces poate avea mai multe thread-uri,
  - thread-urile partajează resursele procesului
  - thread-urile pot comunica foarte rapid între ele prin intermediul memoriei comune fără intervenţia sistemului de operare.
  - crearea unui thread durează mai puţin decât crearea unui proces.
  - un thread se poate bloca fără a bloca activitatea celorlalte thread-uri sau a întregului proces.
  - avantajele thread-urilor pot fi mărite dacă avem un sistem multiprocesor.

Multithreaded programming



Sursa:
http://www.reddit.com/r/aww/comments/2oagj8/multithreaded_programming_theory_and_practice
http://highscalability.com/blog/2014/12/16/multithreaded-programming-has-really-gone-to-the-dogs.html

Acțiuni care afectează execuția thread-urilor:

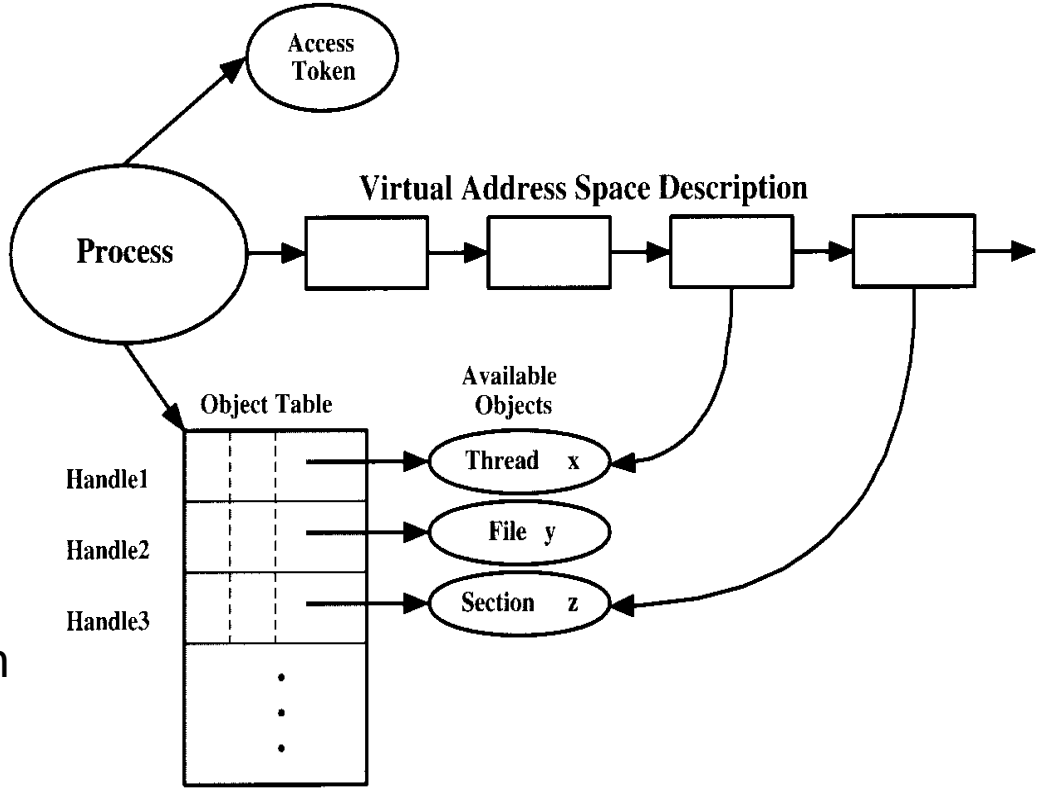
- suspendarea unui proces implică suspendarea tuturor thread-urilor acelui proces deoarece thread-urile partajează acelaşi spaţiu de adrese;
- terminarea procesului atrage terminarea tuturor thread-urilor.
- execuţia unui thread poate fi terminată înaintea terminării execuţiei procesului.

Studii de caz: Windows 2000

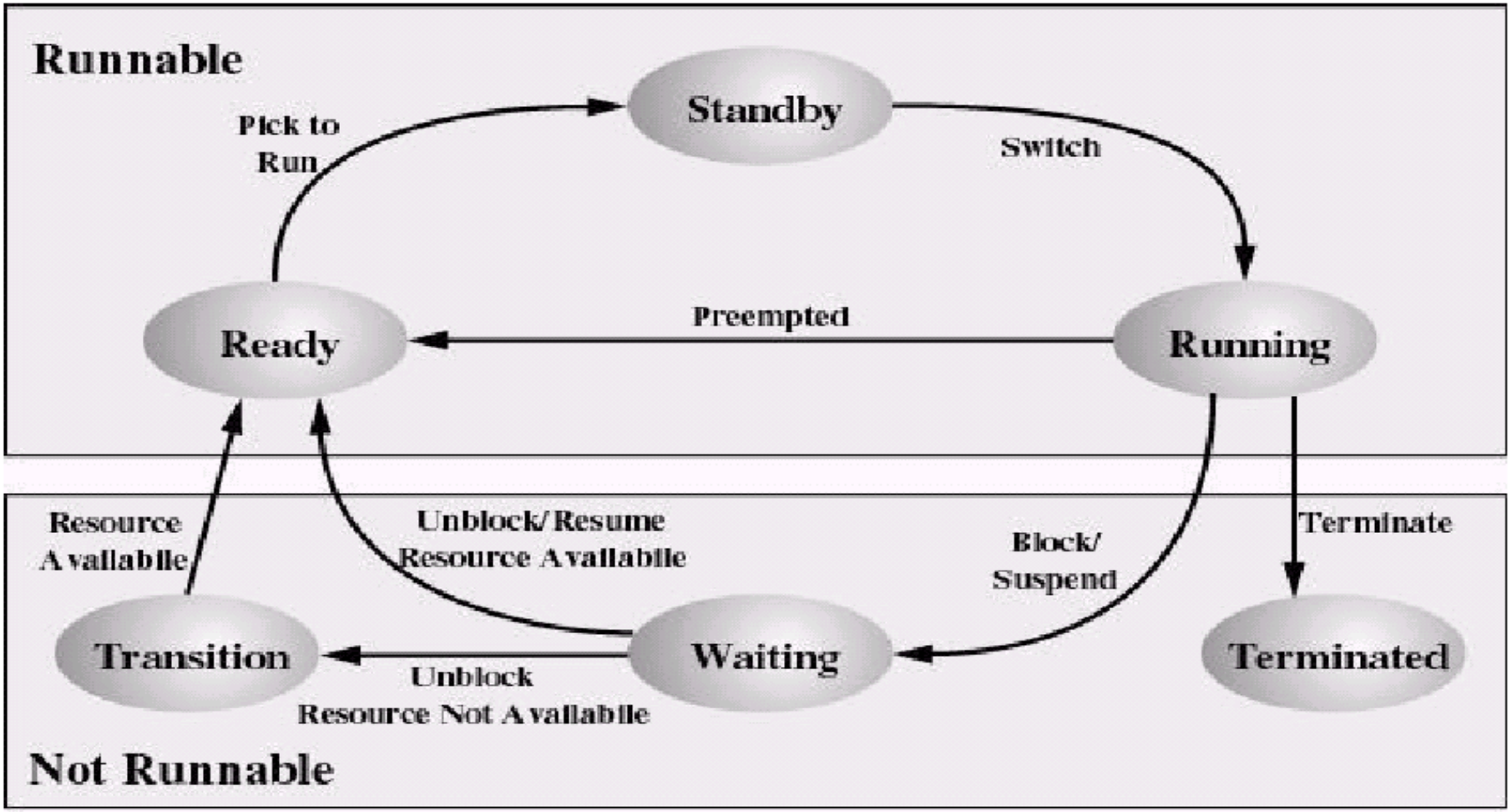
- Caracteristicile proceselor :
  - sunt implementate ca obiecte ;
  - pot conţine mai multe thread-uri ;
  - posibilităţi de sincronizare atât a proceselor, cât şi a thread-urilor.

Windows 2000

- Componentele unui proces :
  - access token : folosit la validarea accesului utilizatorului ;
  - spaţiul virtual de adrese
  - object table : conţine informaţii despre obiectele pe care le deţine procesul
  - fiecare thread are propriul său handler (descriptor – un număr care face referinţă la un obiect).



Windows 2000
Stările unui thread



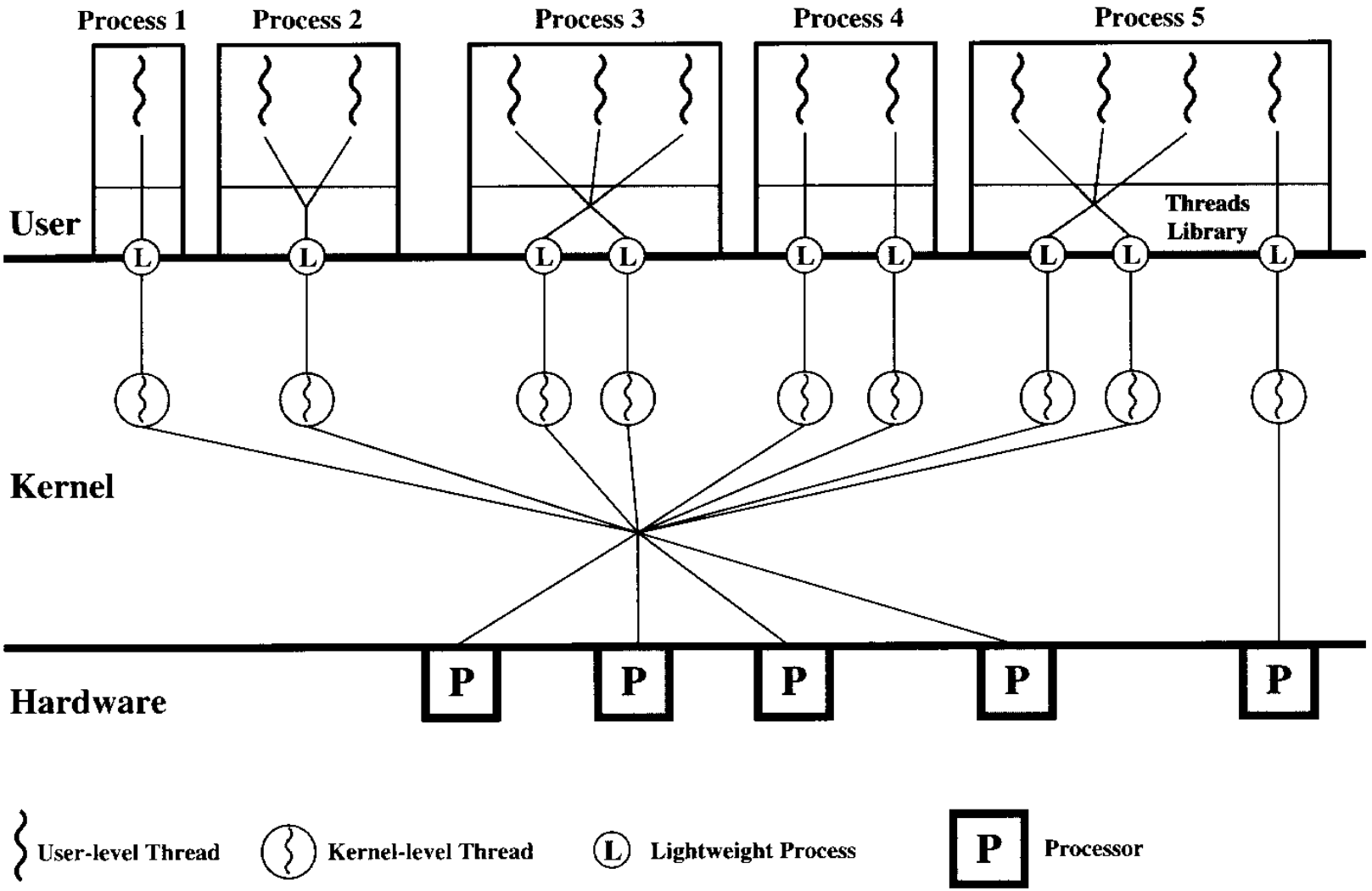
Solaris

- Suportul pentru thread-uri este proiectat astfel încât să ofere o mare flexibilitate în exploatarea resurselor procesorului
- Concepte folosite de Solaris :
  - Process: proces Unix normal care include spaţiul de adrese utilizator, stiva şi PCB (process control block).
  - User-level threads:
    - Implementerea thread-urilor prin intermediul unei biblioteci în spaţiul de adrese al procesului, invizibile sistemului de operare.
    - Implementarea la nivel utilizator reprezintă interfaţa pentru paralelismul aplicaţiilor.

Solaris (2)

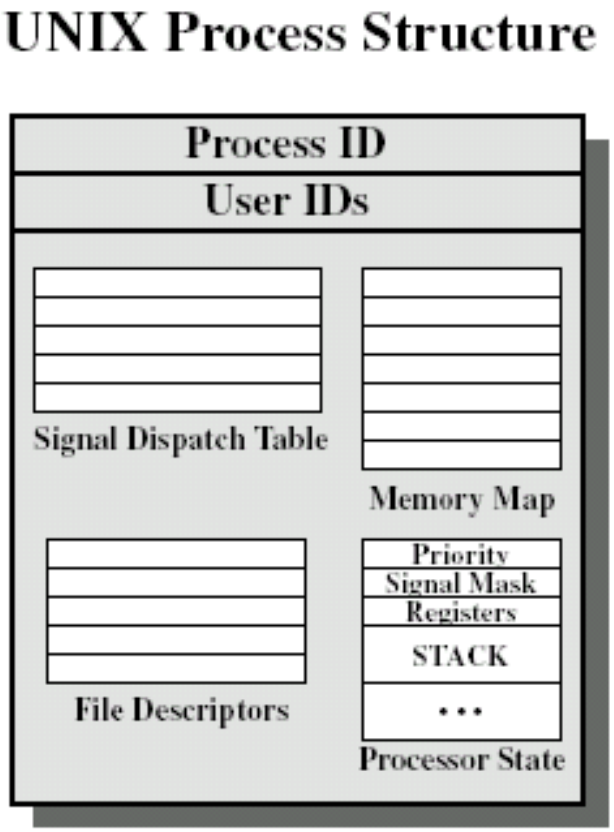
- Concepte folosite de Solaris (2):
  - Lightweight processes (LWP):
    - aceste procese pot fi văzute ca un intermediar între thread-urile utilizator şi thread-urile kernel.
    - Fiecare astfel de proces suportă unul sau mai multe thread-uri utilizator care vor corespunde unui thread kernel.
    - Aceste procese sunt planificate independent de către sistemul de operare şi pot rula în paralel pe mai multe procesoare.
  - Kernel threads:
    - sunt entităţi fundamentale şi sunt planificate şi lansate în execuţie pe unul din procesoarele sistemului.

Solaris - Arhitectura multithreading



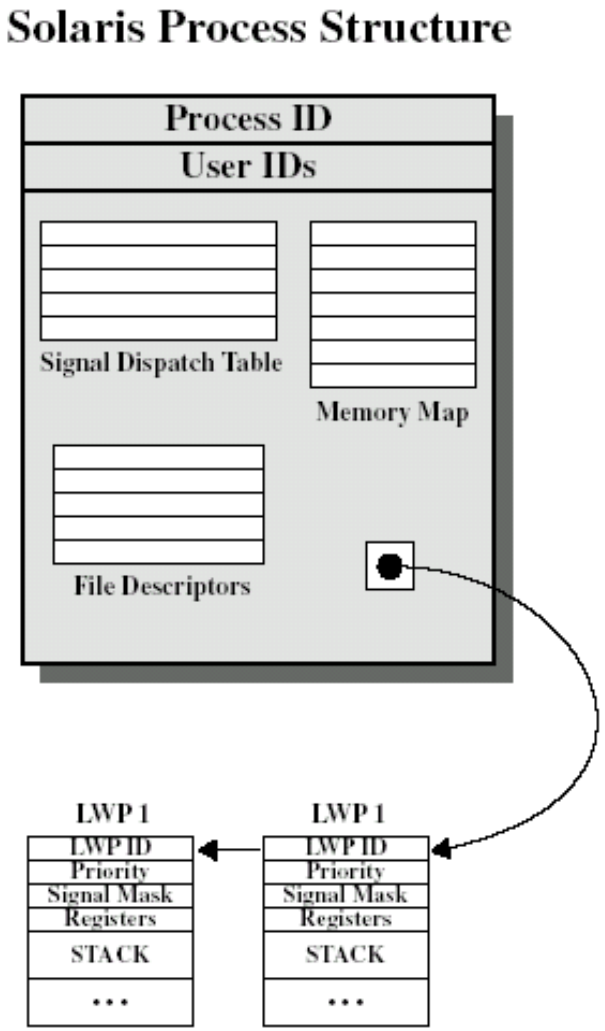
Solaris – Structura proceselor

- sistem Unix structura proceselor include:
  - processor ID
  - user ID
  - tabela de transmitere a semnalelor (pe care kernelul o foloseşte când decide ce să facă cind trimite un semnal unui proces)
  - desciptorii de fişier (descriu starea fişierelor utilizate de un proces)
  - harta memoriei (informaţii privind spaţiul de adrese pentru procese)
  - o structură privind starea procesorului (processor state structure - include stiva kernel pentru proces).

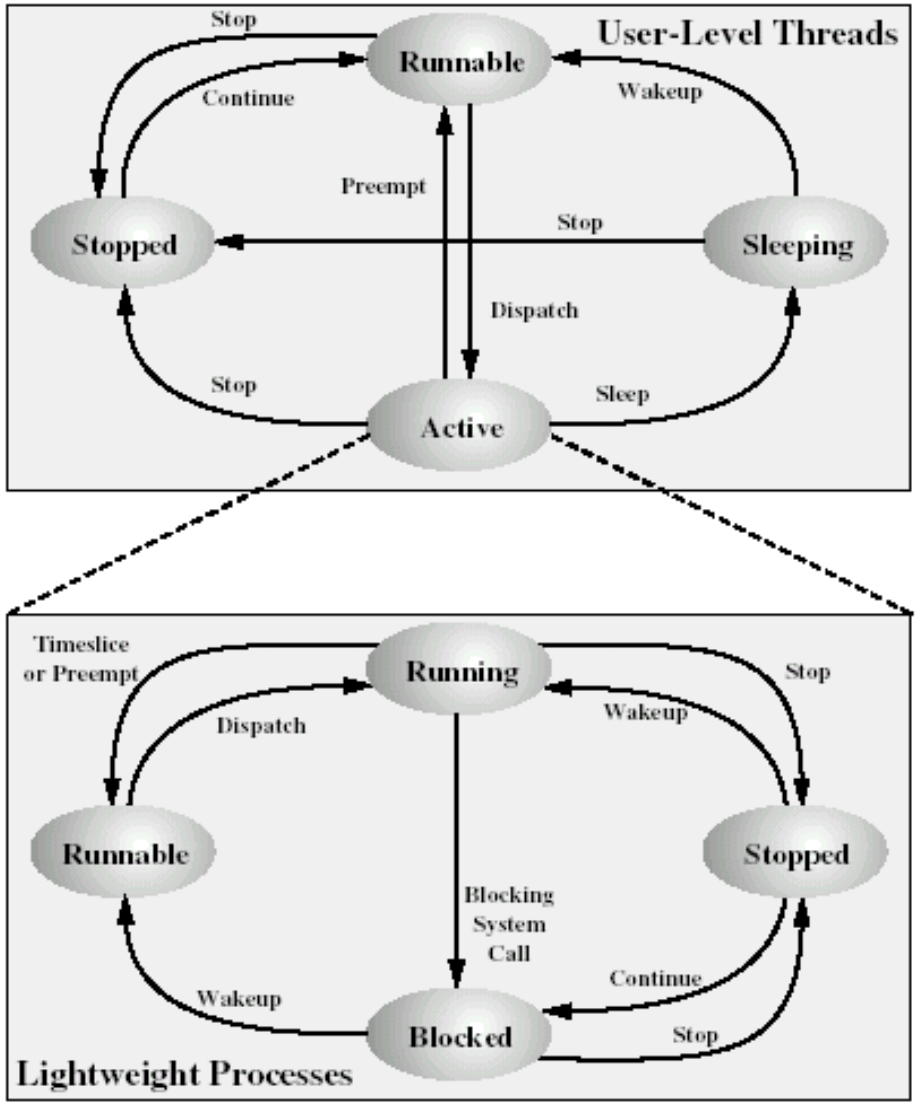


Solaris – Structura proceselor

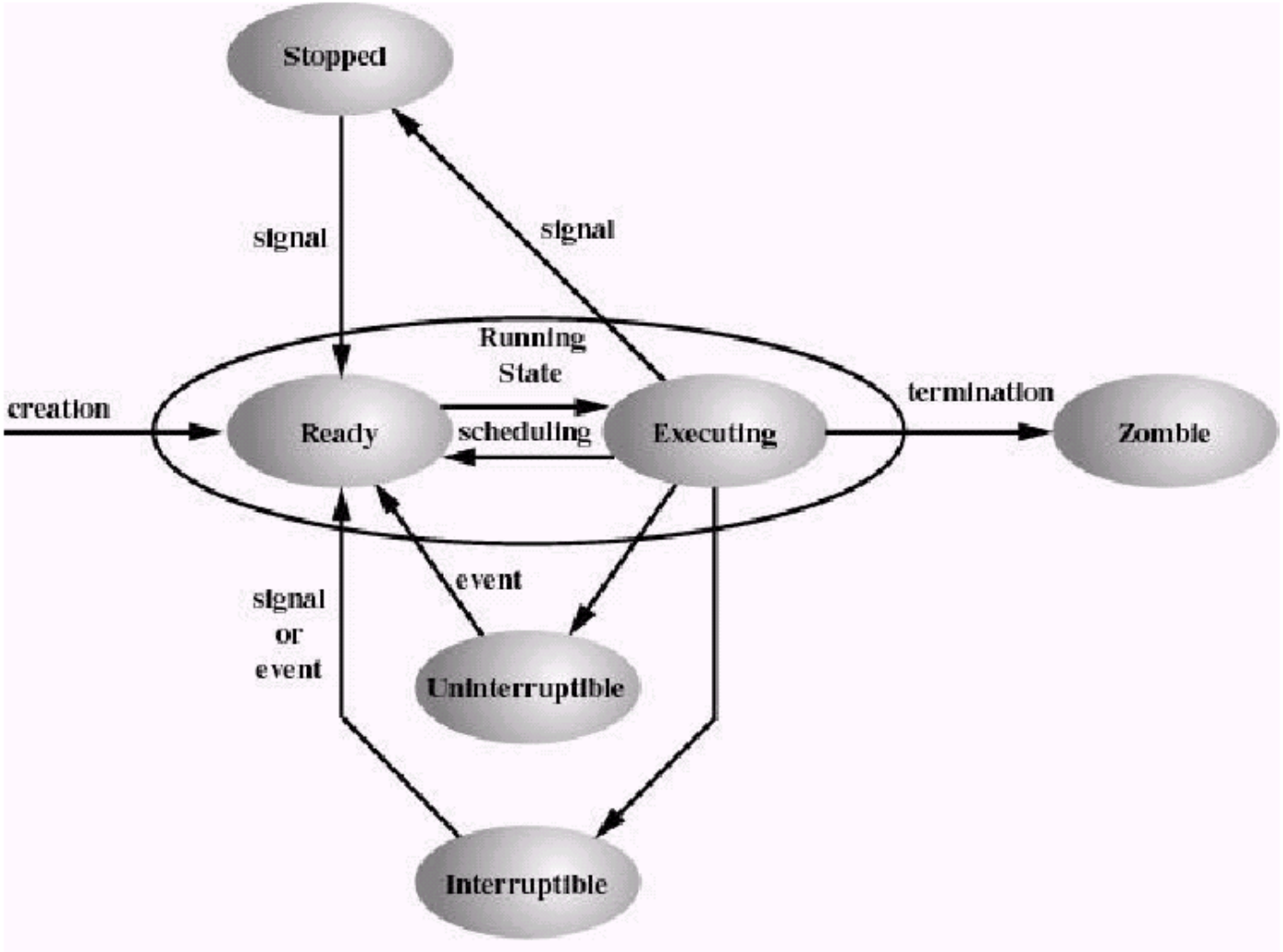
- Se înlocuieşte processor state structure cu o listă de structuri ce conţin cîte un bloc pentru fiecare LWP.
- Structura unui astfel de bloc este următoarea:
  - LWP identifier
  - Prioritatea LWP şi de aici cea a thread-ului kernel corespunzător
  - o mască de semnale care spune kernelului ce semnale vor fi acceptate
  - valorile salvate ale registrelor de la nivel utilizator atunci cînd LWP nu rulează
  - o stivă kernel pentru LWP care include argumentele apelurilor sistem,
  - rezultatele şi codurile de eroare pentru fiecar nivel de apelare.
  - resursele utilizate
  - pointer către thread-ul kernel corespunzător
  - pointer către structura procesului



Solaris - Stările thread-urilor utilizator şi LWP



Linux – Stările proceselor



Linux - task_struct

- State:
  - starea execuţiei unui proces (executing, ready, suspended, stopped, zombie).
- Informaţii de planificare:
  - informaţii folosite de Linux pentru planificarea proceselor.
  - Un proces poate fi normal sau real time şi are o prioritate.
  - Procesele real-time sunt planificate înanintea celor normale şi în cadrul fiecărei categorii există ierarhii de prioritate.
  - Un contor ţine minte timpul în care un proces este executat.
- Identificatori:
  - Fiecare proces are un identificator unic pentru utilizator şi grup.
  - Comunicaţia interprocese:
  - Linux suportă mecanisme de comunicaţie interproces.
- Legături (Links):
  - Fiecare proces păstrează legături către procesul părinte, procesele care au acelaşi părinte cu el şi către procesele fiu.
- Informaţii despre timp:
  - Informaţii despre timpul la care a fost creat procesul, timpul procesor consumat..
- File system:
  - pointeri către fişierele deschise de un proces.
- Virtual memory:
  - defineşte memoria virtuală atribuită acelui proces.
- Processor-specific context:
  - informaţii despre regiştri şi stivă

Linux – thread-uri

- Un proces nou este creat prin copierea atributelor procesului curent.
- Un proces nou poate fi "clonat" astfel încât partajează resursele (fişiere, semnale, memoria virtuală).
- Dacă două procese partajează memoria virtuală, ele vor funcţiona ca thread-uri în cadrul unui singur proces.
- **Nu este definită o structură de date separată pentru un thread şi nu există o distincţie clară între proces şi thread.**
- Pentru lucrul cu thread-uri există biblioteca <pthread.h> .