

# Programare concurentă și distribuită - Lab 2

Faculty of Mathematics and Informatics  
Department of Computer Science

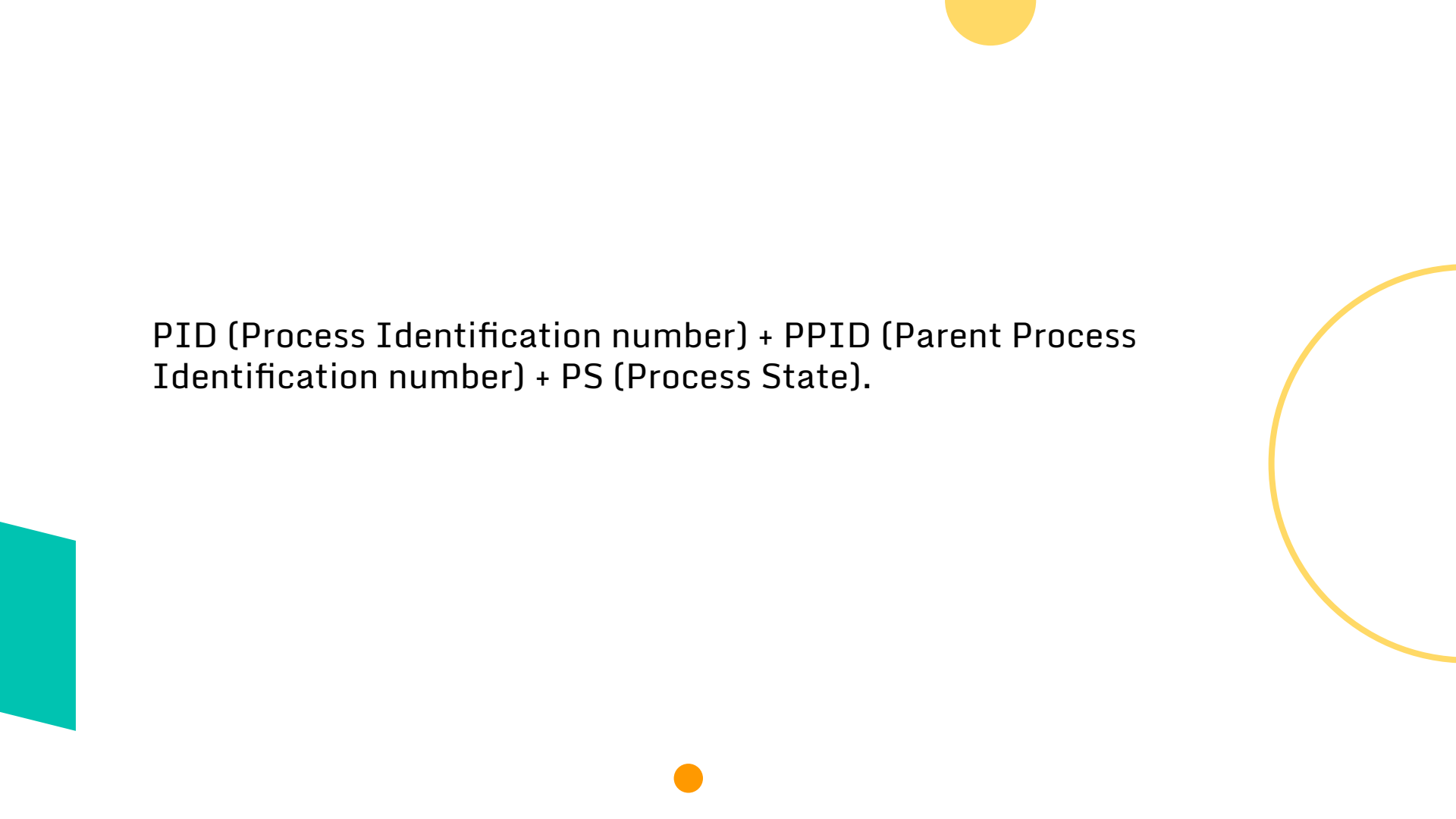
# Conținut

## Laborator 2

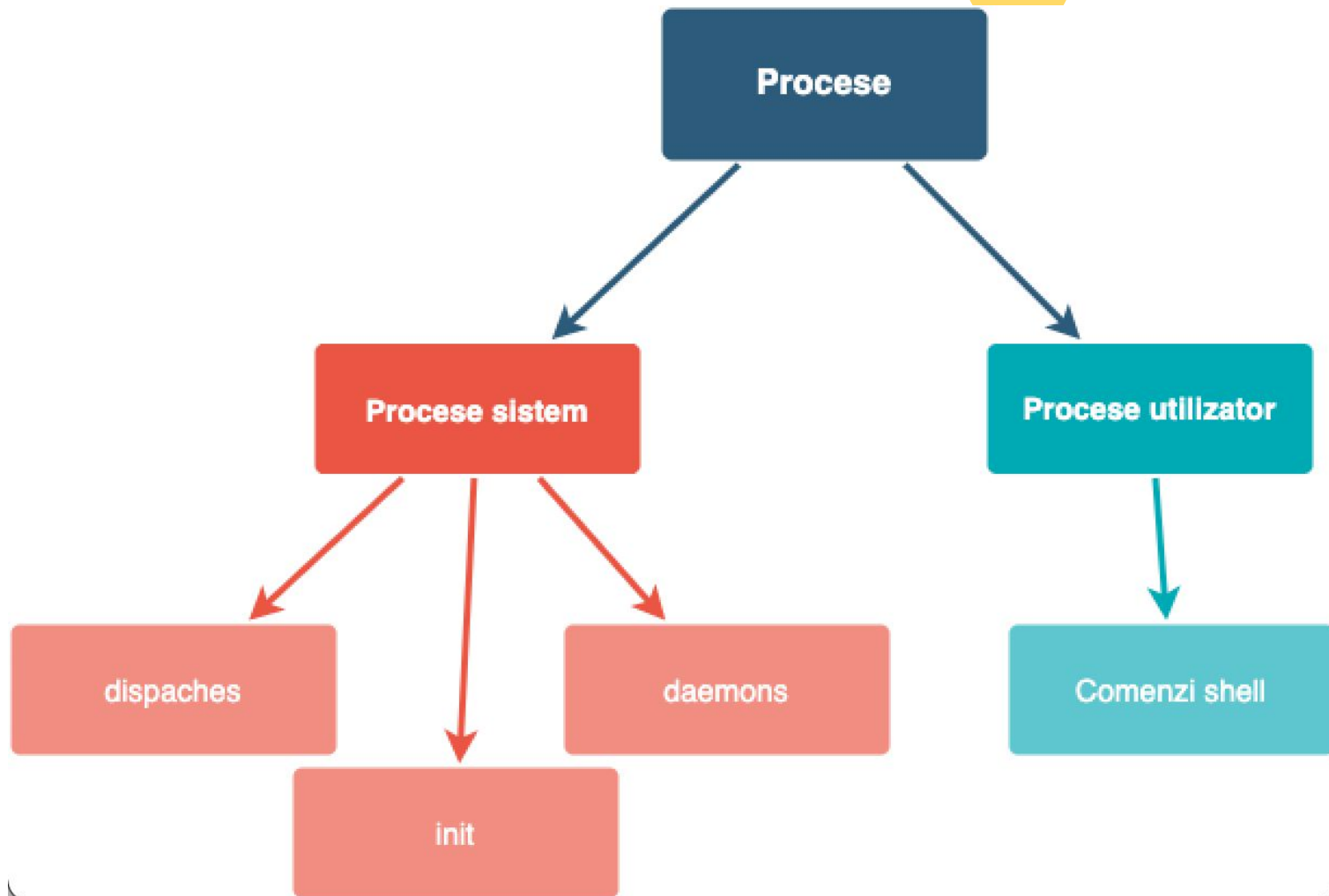
- Procese. Starile unui proces. Executia unui proces.
- Programarea concurenta, paralela si distribuita.
- Crearea si identificarea proceselor
- Netcat

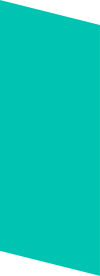



**Ce este un proces?**




PID (Process Identification number) + PPID (Parent Process Identification number) + PS (Process State).





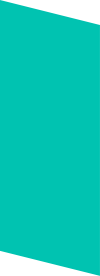




Fiecare proces, la creare, primește din partea sistemului de operare Unix un număr unic de identificare: **PID** (**P**rocess **ID**entification number). La terminarea procesului acest număr este eliberat, el putând fi alocat altui proces.



Fiecare proces aparține unui proprietar: **UID** (**U**ser **ID**entification number); proprietarul unui proces este cel care l-a creat, adică utilizatorul.





Un program poate fi instanțiat de mai multe ori, fiecare instanță constituindu-se într-un proces separat. Fiecărui proces sistemul de operare îi alocă o zonă de memorie separată. Un proces poate “cere”, prin apelul funcției sistem **fork**, crearea unui nou proces.



**Ce este un FORK?**



### Parent

```
main()
{
    pid = 3456
    pid=fork();
    if (pid == 0)
        ChildProcess();
    else
        ParentProcess();
}

void ChildProcess()
{
    .....
}

void ParentProcess()
{
    .....
}
```

### Child

```
main()
{
    pid = 0
    pid=fork();
    if (pid == 0)
        ChildProcess();
    else
        ParentProcess();
}

void ChildProcess()
{
    .....
}

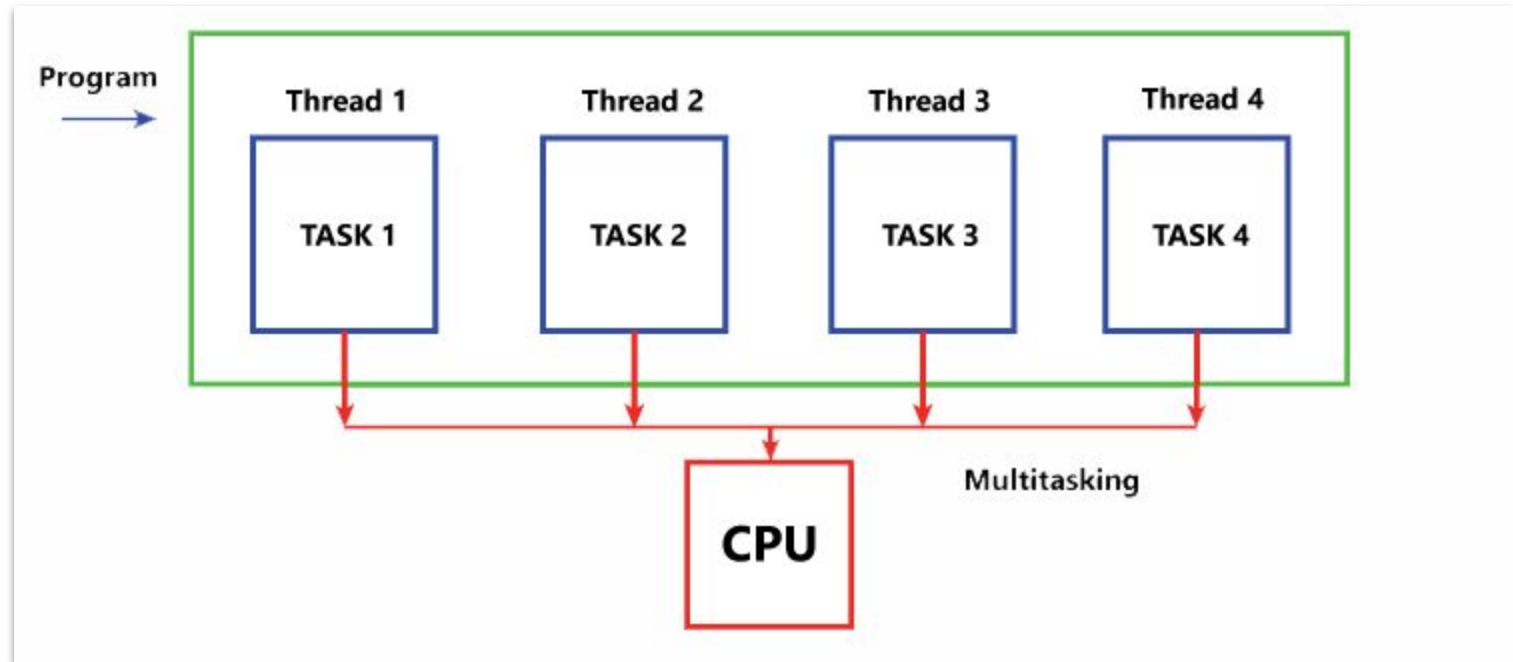
void ParentProcess()
{
    .....
}
```

**La fork cuvântul cheie nu e *furculiță*.  
Cuvântul cheie - bifurcație.**

- 1. Clonezi programul**
- 2. Ai logică separată.**


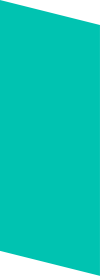



**Ce este un THREAD?**






**Care este diferența dintre un fork și un thread?**




**Diferența - address space**  
**Thread - variabilele rămân aceleași**  
**Fork - variabilele se copiază la copil**

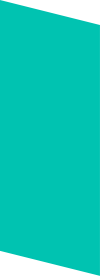






**Ce este programarea concurenta si  
distribuita?**



Programarea concurrent / distribuita = elementul esențial care o deosebește de programarea paralelă este faptul ca procesele cooperează între ele în timpul execuției



Programare paralela = procesele paralele nu sunt condiționate unul de celălalt, nu colaborează între ele, execuția unuia nu este în nici un moment dependentă de rezultatele parțiale ale celuilalt.







# **Crearea și Identificarea proceselor**

## **Creare proces: Funcția fork()**

**Funcția fork() crează un nou proces, identic cu procesul apelant. Noul proces este referit sub numele de copil, iar procesul apelant este referit sub numele de părinte. Cele două procese, părinte și copil, rulează în spații de memorie diferite.**

**Un apel al funcției fork este procesat atât de părinte cât și de copil.**

**Un apel al funcției fork returnează:**

- **0** dacă procesul care procesează apelul este copilul.
- **un întreg pozitiv lung** reprezentând PID-ul copilului dacă procesul care procesează apelul este părintele.
- **-1** dacă apelul funcției sistem fork a eșuat (nu s-a creat copilul)

**Identificare proces: Funcțiile getpid(),  
getppid(), getuid(), geteuid(), getgid() și  
getegid()**



# **Program fork\_.c - creare, terminare și identificare proces**



**<https://ideone.com/y64pW8>**

# **Program process\_chains\_0.c - Creare de procese în lanț**

<https://ideone.com/OjMMi7>





# Program `process_fans.c` - Creare de procese în pieptene





**<https://ideone.com/xNdmNw>**



# Netcat

**Netcat, sau nc în unele implementări UNIX, este un utilitar de rețea utilizat în depanarea și investigarea rețelei. Netcat citește și scrie date în conexiuni de rețea utilizând protocolul TCP/IP sau UDP. Netcat este conceput pentru a fi un instrument de back-end de încredere care poate fi utilizat direct și ușor de alte programe și scripturi. Exemple de utilizare a utilitarului netcat Se deschid două ferestre terminal numite astfel: Fereastra Server + Fereastra Client: Netcat este de fapt utilizat în comunicarea prin socket într-o arhitectură de tip server-client.**