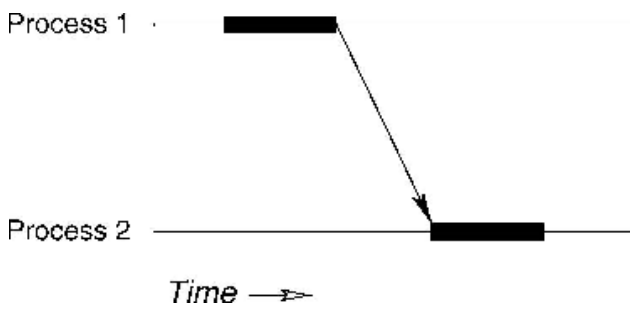
# Sisteme Distribuite – Teorie
# 3. Comunicarea in SD

# SD vs sistem monoprocesor | Stare

- Comunicarea interprocese:
  - Monoprocesor:
    - Comunicarea intre procese presupune implicit ca exista o memorie comuna
    - Ex. Sincronizare – semaforul este partajat
  - SD: nu exista memorie partajata, in loc mesaje
- Procesele: sunt componente active cu state si comportare
  - Starea: consista din datele pe care le administreaza procesul
    - Stare activa sau stare pasiva state or passive state
  - Comportare: implementarea logicii aplicatiilor
- Mesaj: secventa de octeti care sunt transportati intre doua procese via un mediu de comunicare

# Communicarea intre 2 procese

- Unul este expeditorul, altul este destinatorul (receptorul)
- Diagrama spatiu-timp
  - Starea activa:
    - Linie neagra groasa
    - Au loc calcule




- Sabloane in scurgerea mesajelor:
  - Comunicarea orientata spre mesaje
  - Comunicarea orientata spre cereri

- Fara asteptare raspuns

- Cu raspuns de la destinatar

# Sincronicitatea mecanismului de comunicare

- descrie separarea in timp intre expeditor si destinatar
- *Comunicarea sincrona:*
  - Expeditorul este pasiv in timpul procesului de comunicare pana cand mesajul ajunge la destinatar
- *Comunicarea asincrona:*
  - Expeditorul ramane activ dupa ce mesajul a fost expediat
  - Mai rapid! Suporta mai bine paralelismul proceselor
  - Necesar un SD care este capabil sa tina mesaje in tampon

# Clasificarea mecanismelor de comunicare

| *Sablon comunicare* | *Nivele de sincronizare* | |
|---|---|---|
| | *Asincron* | *Sincron* |
| Orientat mesaj | 1/fara asteptare la expediere | 2/rendezvous |
| Request-oriented | 3/invocare de serviciu la distanta | 4/apel de procedura la distanta |

Exemple:

1. Serviciile datagrama:
   - comunicare asincrona orientata spre mesaje;
   - expeditorul trimite un mesaj la destinatar fara a astepta un raspuns sau a schimba starea intr-una pasiva
2. Rendezvous:
   - stabileste un moment (logic) pentru intalnirea intre expeditor si destinatar
3. RSI:
   - Expeditorul ramane activ atata timp cat destinatorul proceseaza cererea
4. RPC:
   - Expeditorul transmite o cerere la destinatar si este pasiv pana cand destinatarul trimite rezultatele cererii

# Modelul client-server

- Avantaje:
  - Simplicitate – nu este necesara stabilirea unei conexiuni anterioare
  - Unul sau mai multi clienti pot avea acces la serviciu
  - Orice comunicare din tabelul anterior
  - Usurinta in migrarea aplicatiilor existente bazate pe programare procedurala
  - Potential pentru concurenta
- Dezavantaje:
  - Restrictionarea la paradigmele programarii procedurale exclude alte abordari precum programarea functionala sau declarativa
  - Transparenta nu mai poate fi atinsa in cazul unui esec radical de sistem
  - Probleme cauzate de concurenta cand procesele trebuie sa se sincronizeze

# Procedurile din biblioteci pentru servicii de comunicare

- *send(dest,&mptr)*
  - Expediaza mesajul referit prin *mptr* la un proces identificat de *dest*
  - Cauzeaza blocarea apelantului pana cand mesajul a fost expediat
- *receive(addr,&mptr)*.
  - Cauzeaza blocarea apelantului pana cand soseste mesajul
  - Cand acesta soseste, este copiat intr-un tampon indicat de referinta *mptr* si apelantul este deblocant
  - Parametrul *addr* specifica adresa la care destinatarul asteapta mesajul
- Sunt posibile numeroase varianate ale acestor proceduri si a parametrilor lor

# Primitive cu blocare

- Primitivele de transmitere de mesaje sunt numite primitive blocante
- Sunt numite si primitive sincrone
- Cand un proces apeleaza send
  - Specifica o destinatie si un tampon care sa fie expediat la destinatie
  - Cand mesajul este expediat, procesul expeditor este suspendat in blocare
  - Instructiunile care urmeaza apelul send-ului nu sunt executate pana cand mesajul n-a fost expediat complet
- Un apel la un receive
  - Nu returneaza controlul pana cand mesajul a fost receptionat si pus intr-un tampon mesaj indicat de parametru
  - Procesul ramane suspendad in receive pana cand vine mesajul chiar daca poate sa tina ore intregi
  - In anumite sisteme, destinatorul poate specifica de la cine doreste sa receptioneze, in care caz ramane blocat pana cand soseste mesajul de la acel expeditor

# Primitive ne-blocante

- Numite si primitive asincrone
- Daca expedierea este ne-blocanta, se returneaza controlul la apelant imediat, inainte de expedierea mesajului
- Avantajul acestei scheme este aceea ca procesul de expedoere poate continua calculul in paralel cu transmiterea mesajului, in loc sa aiba CPU-l inactiv
- Alegerea intre primitivele blocante si cele neblocate este in mod normal realizata de designerii de sistem

# Probleme cu primitivele ne-blocante

- Expeditorul nu poate modifica tamponul mesajului pana cand mesajul a fost expediat
- Procesul expeditor nu are nici o idee cand are loc transmiterea, astfel incat nu stie cand este sigure sa foloseasca tamponul
- Solutii:
  - Copiaza mesajul intr-un tampon intern si permite procesului sa continue
    - Seamana cu un apel blocant: cand primeste controlul inapoi este liber sa utilizeze tamponul
    - Dar mesajul nu este inca transmis si expeditorul nu este impiedicat de acest fapt
    - Dezavantajul acestei metode este acela ca toate mesajele trebuie copiate din spatiul utilizator in spatiul nucleului
  - Intrerupe expeditorul cand mesajul a fost expediat pentru a-l informa ca tamponul este din nou disponibil
    - Intreruperile la nivel utilizator fac programarea dificila si subiect pentru conditiile de intrecere care pot conduce la imposibilitatea de repetare
    - Metoda este eficienta si permite paralelismul
    - Dezavantaj: programele bazate pe intreruperi sunt dificil a fi scrise corect si aproape imposibil sa fie depanate cand ceva nu merge bine

# Puncte de vedere

- Al designerului de sistem de operare:
  - Diferente intre primitivele sincone si asincrone: expeditorul poate utiliza sau nu tamponul pentru mesaje imediat dupa re-obtinerea controlului
- Al designerului de limbaje de programare:
  - Sincron: expeditorul este blocat pana cand destinatarul a acceptat mesajul si confirmarea a fost primita de catre expeditor
  - Orice altceva este asincron
  - Daca expeditorul obtine controlul inapoi inainte ca mesajul sa fie copiat sau expediat, primitiva este asincrona
  - Cand expeditorul este blocat pana cand destinatarul a confirmat mesajul, este vorba de o primitiva asincrona

# Receptionare ne-blocanta

- Returneaza controul aproape imediat
- Cand stie apelantul ca operatia a fost terminata?
  - Oferirea unei primitive explicite care permite destinatarului sa fie blocat cand este necesar
  - Ofera o primitiva de testare pentru a permite destinatarului sa solicite nucleului sa verifice starea
    - Exemplu: receptionare conditionata, care fie returneaza un mesaj sau semnaleaza esec, dar in orice caz returneaza imediat sau intr-un interval dat.

# Time-out

- Intr-un sistem in care apelurile send sunt blocante, expeditorul se poate bloca pentru totdeauna
- Pentru a preveni aceasta situatie, in anumite sisteme apelentul poate specifica un interval de timp in care asteapta un raspuns
- Daca nu primeste nici un raspuns in respectivul interval, apelul send se termina cu un status de eroare