# Операційні системи

Лекція 5

Керування процесами і потоками

# План лекції

- Мультипрограмування
- Означення процесу і потоку
- Моделі процесів і потоків
- Керування потоками, планування
- Опис процесів і потоків: керуючий блок, образ, дескриптор і контекст
- Стани потоків

# Мультипрограмування

- *Мультипрограмування*, або *багатозадачність* (*multitasking*) – спосіб організації обчислювального процесу, в якому на одному процесорі почергово виконуються кілька програм
- Критерії ефективності обчислювальної системи:
  - *Перепускна спроможність* – кількість задач, яку здатна ефективно виконувати система в одиницю часу
  - Зручність роботи користувачів (насамперед, можливість одночасно інтерактивно працювати з низкою прикладних програм на одній машині)
  - *Реактивність системи* – здатність системи дотримувати наперед задані (короткі) інтервали між одержанням запиту і результатом його оброблення

# Процеси і потоки

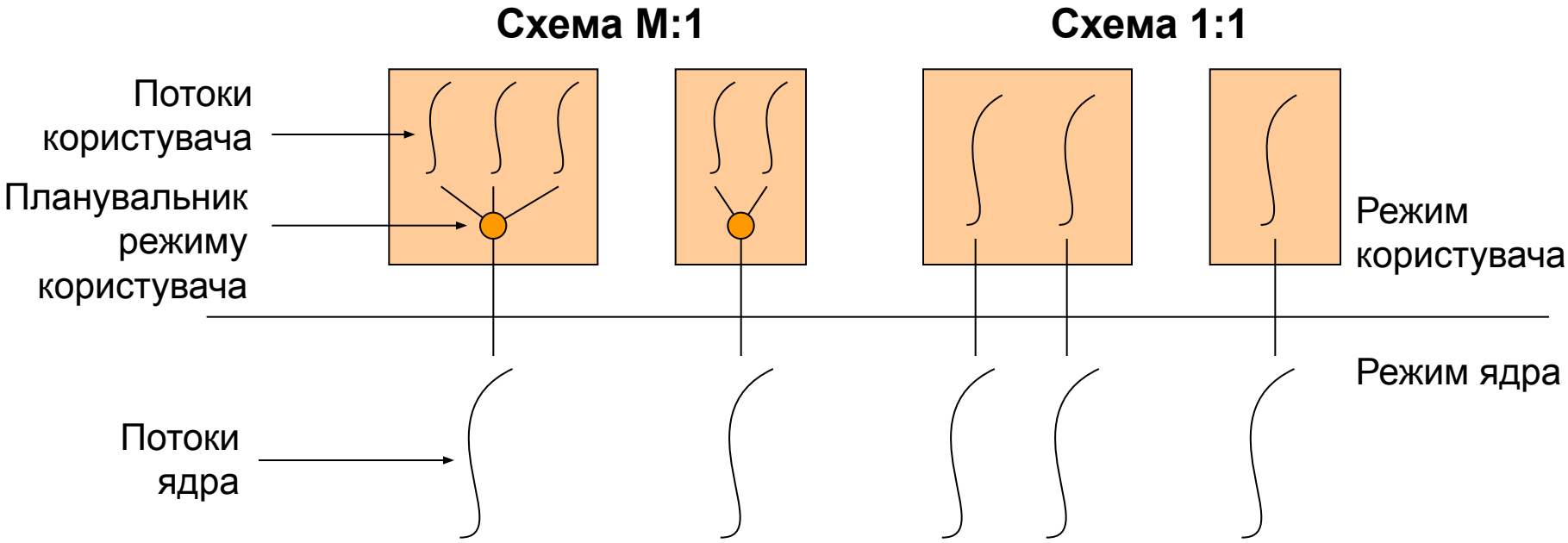
- *Процес* – це абстракція, що описує програму, яка виконується в даний момент
- Складові частини процесу:
  - Послідовність виконуваних команд процесора
  - Набір адрес пам'яті (адресний простір), у якому розташовані команди процесора і дані для них
- *Потік* (*нитка*, *thread*) – набір послідовно виконуваних команд процесора
- У багатопотокових системах процес – це сукупність одного або декількох потоків і захищеного адресного простору, у якому ці потоки виконуються

# Моделі процесів і потоків

- Однозадачні системи – один адресний простір (один процес), у якому може виконуватись один потік
- Деякі сучасні вбудовані системи – один адресний простір (один процес), у якому можуть виконуватись кілька потоків
- Однопотокова *модель процесів* (традиційні системи UNIX) – багато процесів, але у кожному з них лише один потік
- Багатопотоковість, або *модель потоків* (більшість сучасних ОС) - багато процесів, у кожному з яких може бути багато потоків

# Потоки ядра і потоки користувача

- Ядро ОС керує потоками ядра. Керування потоками користувача здійснюють спеціальні системні бібліотеки (підтримка *потоків POSIX*)
- Існують різні *схеми багатопотоковості*:



# Переваги і недоліки багатопотоковості

\+ (переваги)

- Реалізація різних видів *паралелізму* (*concurrency*) – багатопроцесорних обчислень, введення-виведення, взаємодії з користувачем, розподілених застосувань
- Масштабованість (особливо із зростанням кількості процесорів)
- Необхідно менше ресурсів, ніж для підтримки процесів
- Ефективний обмін даними через спільну пам'ять

– (недоліки)

- Складність розроблення й налагодження багатопотокових застосувань
- Зниження надійності застосувань (можливі “гонки”, витоки пам'яті, втрата даних)
- Можливість зниження продуктивності застосувань

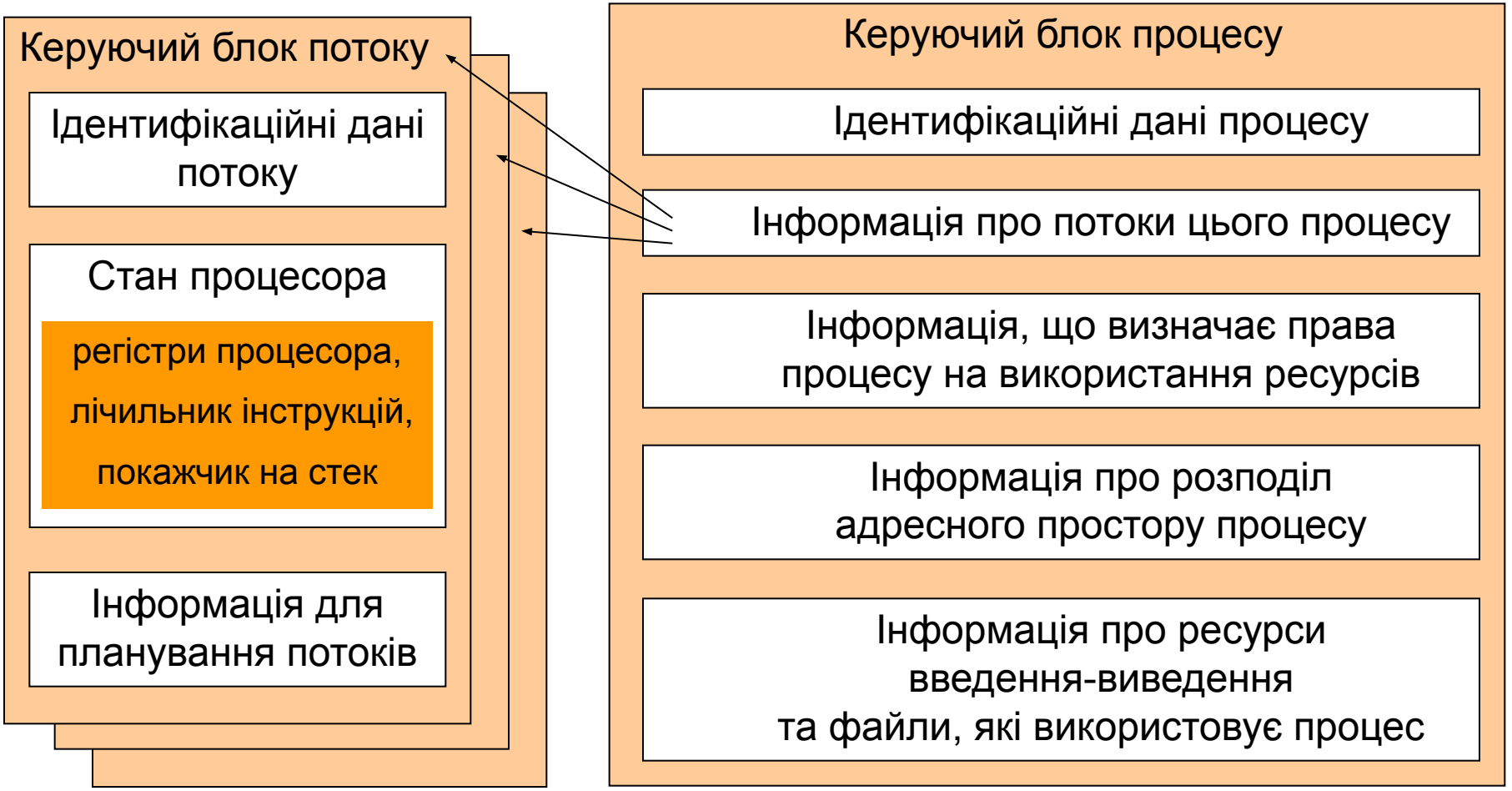
# Завдання підсистеми керування процесами (потоками)

- Створення та знищення процесів і потоків
- *Планування* виконання процесів або потоків, тобто розподіл процесорного часу між ними
  - Визначення моменту часу для зміни потоку, що виконується
  - Вибір наступного потоку для виконання
  - Переключення контекстів
- Забезпечення процесів і потоків необхідними ресурсами
- Підтримання взаємодії між процесами

# Опис процесів і потоків

- *Керуючий блок потоку* (*Thread Control Block*, TCB)
  - Ідентифікаційні дані потоку
  - Стан процесора потоку (регістри процесора, лічильник інструкцій, покажчик на стек)
  - Інформація для планування потоків
- *Керуючий блок процесу* (*Process Control Block*, PCB)
  - Ідентифікаційні дані процесу
  - Інформація про потоки цього процесу (наприклад, покажчики на їхні керуючі блоки)
  - Інформація, на основі якої можна визначити права процесу на використання ресурсів
  - Інформація про розподіл адресного простору процесу
  - Інформація про ресурси введення-виведення та файли, які використовує процес
- *Таблиця процесів* (потоків) – зв'язний список або масив відповідних керуючих блоків

# Опис процесів і потоків



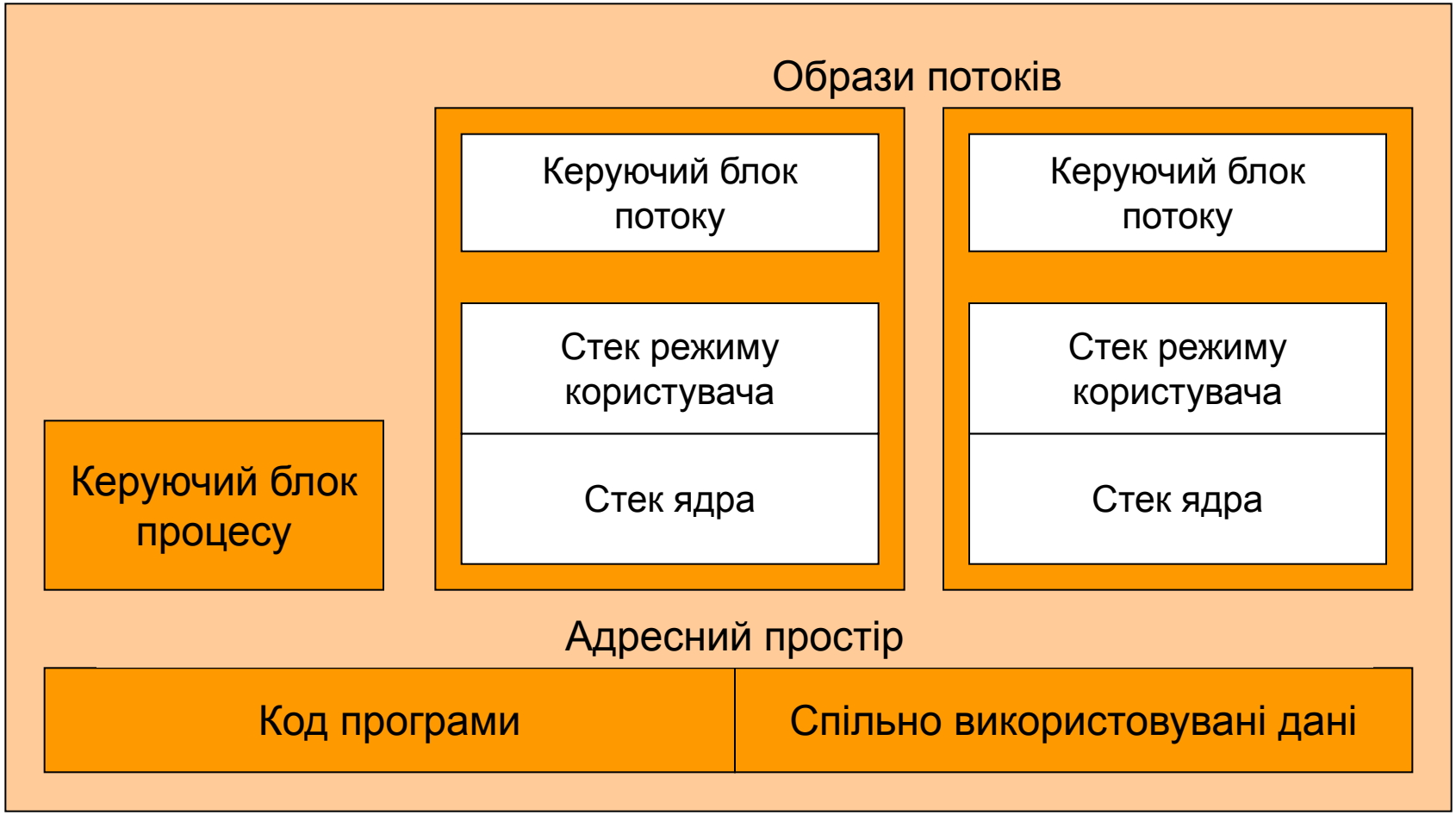
# Приклади структур, що описують процеси

- OS/360 – Task Control Block, TCB (керуючий блок задачі)
- OS/2 – Process Control Block, PCB (керуючий блок процесу)
- UNIX – proc, дескриптор процесу
- Windows – object-process (об'єкт-процес)

# Образи процесу і потоку

- Образ потоку
  - Керуючий блок потоку
  - Стек ядра
  - Стек користувача
- Образ процесу
  - Керуючий блок процесу
  - Програмний код користувача
  - Дані користувача
  - Інформація образів потоків процесу

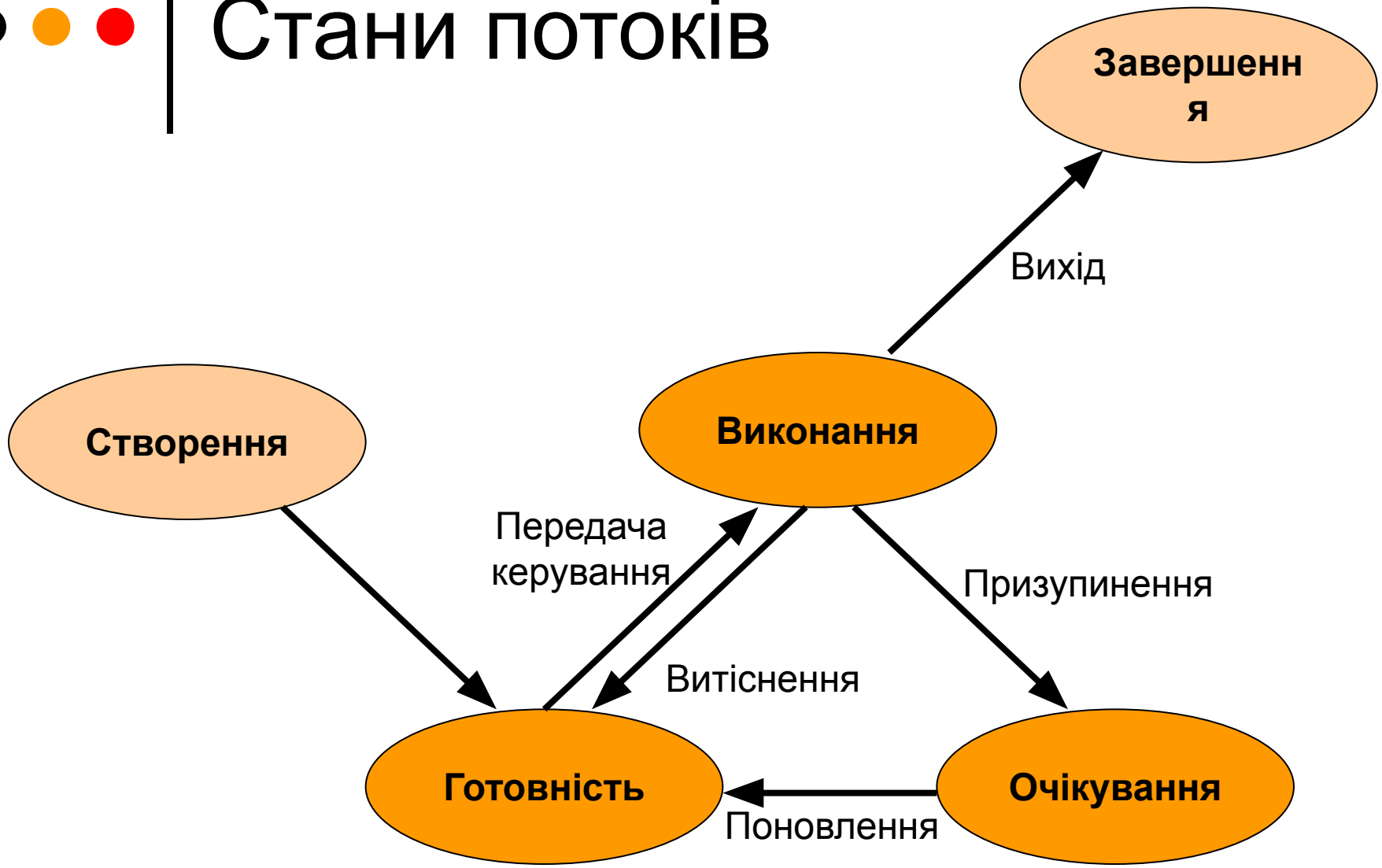
# Образ процесу



# Дескриптор і контекст процесу (UNIX)

- *Дескриптор* – структура, що містить інформацію, необхідну для планування процесів. Ця інформація необхідна протягом усього часу життєвого циклу процесу
  - Ідентифікатор процесу
  - Стан процесу
  - Пріоритет процесу
  - Розміщення процесу в оперативній пам'яті і на диску
  - Ідентифікатор користувача, що створив процес
- *Контекст* – структура, що містить інформацію, необхідну для відновлення виконання процесу (менш оперативна, але більш об'ємна частина інформації, ніж у дескрипторі)
  - Стан регістрів процесора
  - Коди помилок виконаних системних викликів
  - Інформація про відкриті файли і незавершені операції введення-виведення
- Таблиця процесів – список дескрипторів
- Контекст зв'язаний з образом процесу і переміщується разом з ним (наприклад, може бути витісненим з оперативної пам'яті на диск)

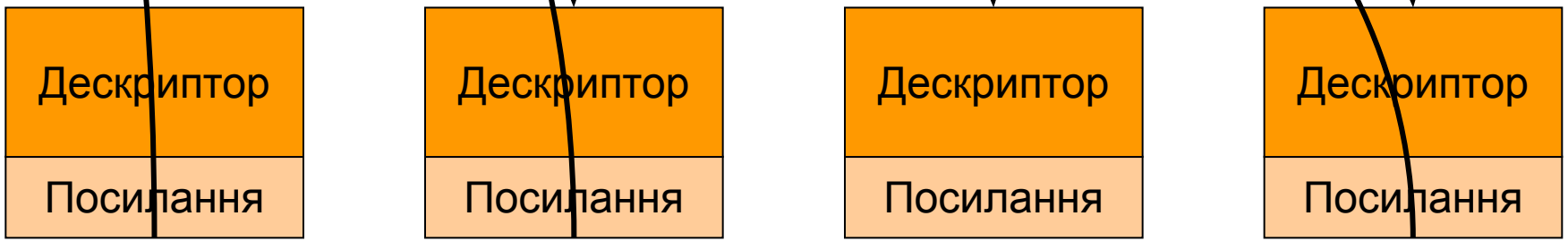
# Стани потоків



# Створення процесів

- Необхідні дії
  - Створити інформаційні структури, що описують процес (керуючий блок: дескриптор, контекст)
  - Виділити оперативну пам’ять
  - Завантажити кодовий сегмент процесу в оперативну пам’ять
  - Поставити дескриптор процесу у чергу готових процесів
- Створення у два етапи (POSIX)
  - fork() – створює точну копію поточного процесу
  - exec() – заміняє код поточного процесу на код іншого
- Створення в один етап (Windows) – CreateProcess()
  (це не системний виклик, а бібліотечна функція)
- Копіювання під час запису
  - Під час виклику fork() дані з пам'яті предка у пам’ять нащадка не копіюють, а натомість відображають адресний простір і помічають області пам'яті як захищені від запису
  - У разі спроби запису виділяють пам’ять і здійснюють копіювання

# Черги процесів (потоків)



# Керування процесами в UNIX/Linux

- Образ процесу містить:
  - Керуючий блок
  - Код програми
  - Стек процесу
  - Глобальні дані
- Ідентифікатор процесу PID – унікальний
- Підтримується зв’язок предок-нащадок
  - Ідентифікатор процесу-предка PPID вказують під час створення
  - Якщо предок завершує роботу, то PPID:=1 (процес init)
- Створення процесу: fork(), exec() (Linux – execve())

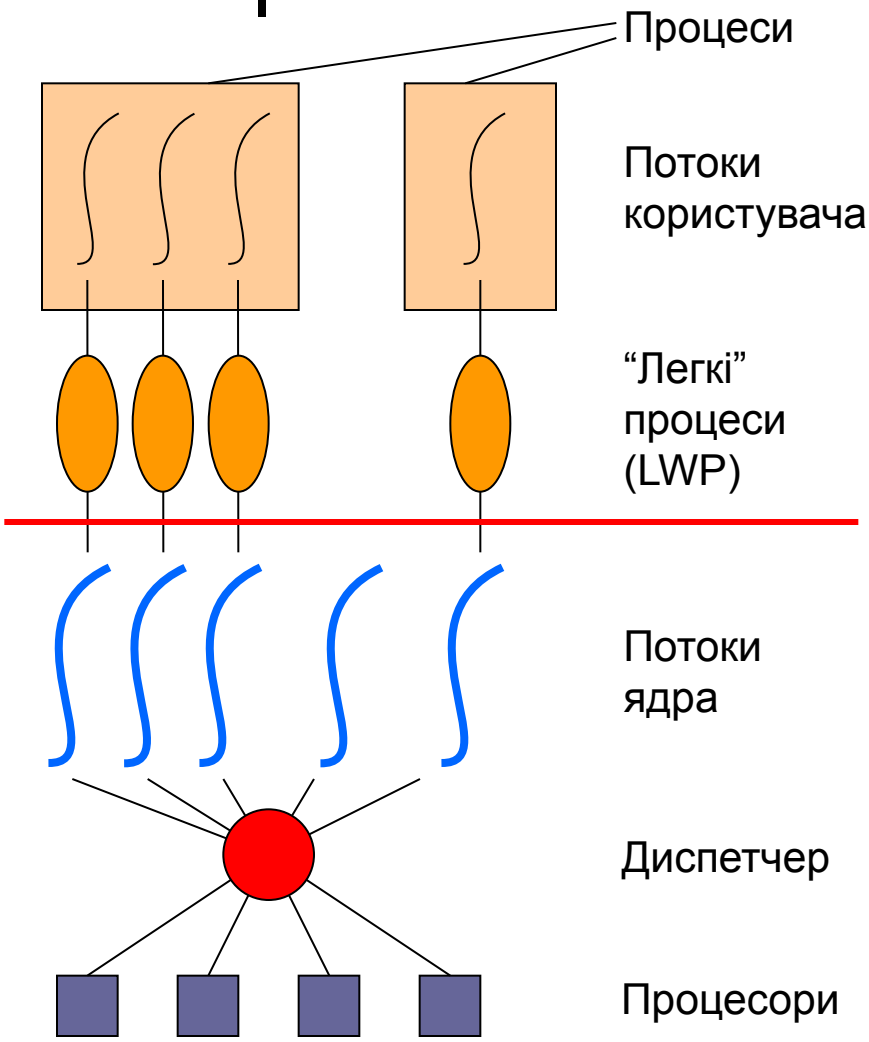
# Керування процесами в UNIX/Linux

- Керуючий блок в Linux – структура task_struct, містить в собі і дескриптор, і контекст
  - В ядрі до версії 2.4 керуючі блоки зберігались у масиві максимального розміру 4 кБ (“таблиця процесів системи”)
  - В ядрі версії 2.4 і вище – дві динамічні структури без обмеження довжини
    - Хеш-таблиця (дає змогу швидко знаходити процес за його PID)
    - Кільцевий двозв’язний список, що забезпечує виконання дій у циклі для усіх процесів системи
- У UNIX SVR4 і BSD керуючий блок складається з двох структур
  - proc – дескриптор (утворюють таблицю процесів)
  - user (u) – контекст (пов'язаний з процесом)

# Багатопотоковість у Linux

- Традиційна реалізація: бібліотека LinuxThreads
  - Потоки – це процеси, що користуються спільними структурами даних, але мають окремі стеки
  - Створення потоку: системний виклик clone()
  - Недоліки:
    - Створення потоку збільшує кількість процесів у системі
    - Кожний потік має власний PID (це суперечить POSIX)
    - Існує зв’язок предок-нащадок (чого не повинно бути)
    - Кожне багатопотокове застосування обов'язково створює додатковий потік-менеджер
- Нова реалізація: NPTL (Native POSIX Threads Library), що спирається на нові функціональні можливості ядра
  - Як процес у системі реєструють лише перший потік застосування
  - Усі потоки процесу повертають один і той самий PID
  - Зв’язок предок-нащадок між потоками не підтримується
  - Потік-менеджер не потрібен
  - Зняті обмеження на кількість потоків у системі
- У більшості UNIX-систем, на відміну від Linux, реалізована повна підтримка POSIX Threads

# Багатопотоковість у Solaris




- Модель багатопотоковості – 1:1 (починаючи з Solaris 8)
- Кожен потік користувача відповідає одному окремому потоку ядра (Kernel Thread, kthread)
- Фактично планування здійснюється для потоків ядра – це найголовніша відмінність від традиційної системи UNIX
- Кожен потік ядра в процесі повинен мати власний “легкий” процес (Lightweight process, LWT) – віртуальне середовище виконання, що має власний стек

# Керування процесами у Windows

- Адресний простір процесу складається з набору адрес віртуальної пам'яті, які цей процес може використовувати
  - Адреси можуть бути пов'язані з оперативною пам'яттю, а можуть – з відображеними у пам’ять ресурсами
  - Адресний простір процесу недоступний іншим процесам
- Процес володіє системними ресурсами, такими як файли, мережні з'єднання, пристрої введення-виведення, об’єкти синхронізації
- Процес не має прямого доступу до процесора
- Процес містить стартову інформацію для потоків, які у ньому створюються
- Процес обов'язково має містити хоча б один потік

# Структури даних процесу у Windows

- Доступні лише у привілейованому режимі:
  - Для виконавчої системи – *об'єкт-процес виконавчої системи* (*Executive Process Block*, *EPROCESS*)
    - *KPROCESS*
    - Ідентифікаційна інформація (PID, PPID, ім'я файлу)
    - Інформація про адресний простір процесу
    - Інформація про ресурси, доступні процесу
    - *PEB*
    - Інформація для підсистеми безпеки
  - Для ядра – *об'єкт-процес ядра* (*Kernel Process Block*, *KPROCESS*)
    - Покажчик на ланцюжок блоків потоків ядра
    - Інформація, необхідна ядру для планування
- Доступна у режимі користувача:
  - *Блок оточення процесу* (*Process Environment Block*, *PEB*)
    - Початкова адреса ділянки пам'яті, куди завантажився програмний файл
    - Покажчик на динамічну ділянку пам'яті, що доступна процесу

# Керування потоками у Windows

- Елементи потоку:
  - Вміст набору регістрів, який визначає стан процесора
  - Два стеки (для режиму ядра і режиму користувача), що розміщені в адресному просторі процесу
  - Локальна пам’ять потоку (TLS)
  - Унікальний ідентифікатор потоку TID
- Розрізняють потоки ядра і потоки користувача

# Структури даних потоку у Windows

- Доступні лише у привілейованому режимі:
  - Для виконавчої системи – *об'єкт-потік виконавчої системи* (*Executive Tread Block*, *ETHREAD*)
    - *KTHREAD*
    - Ідентифікаційна інформація (*PID*, покажчик на *EPROCESS*)
    - Стартова адреса потоку
    - Інформація для підсистеми безпеки
  - Для ядра – *об'єкт-потік ядра* (*Kernel Thread Block*, *KTHREAD*)
    - Покажчик на стек режиму ядра
    - Покажчик на *TEB*
    - Інформація, необхідна ядру для планування
    - Інформація, необхідна для синхронізації потоку
- Доступна у режимі користувача:
  - *Блок оточення потоку* (*Thread Environment Block*, *TEB*)
    - Ідентифікатор потоку *TID*
    - Покажчик на стек режиму користувача
    - Покажчик на *PEB*
    - Покажчик на локальну пам’ять потоку