Робоча програма дисципліни «Системне програмне забезпечення»

СПЕЦІАЛЬНІСТЬ 123 · КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

БАКАЛАВР

120

Годин

Загальний обсяг дисципліни

4

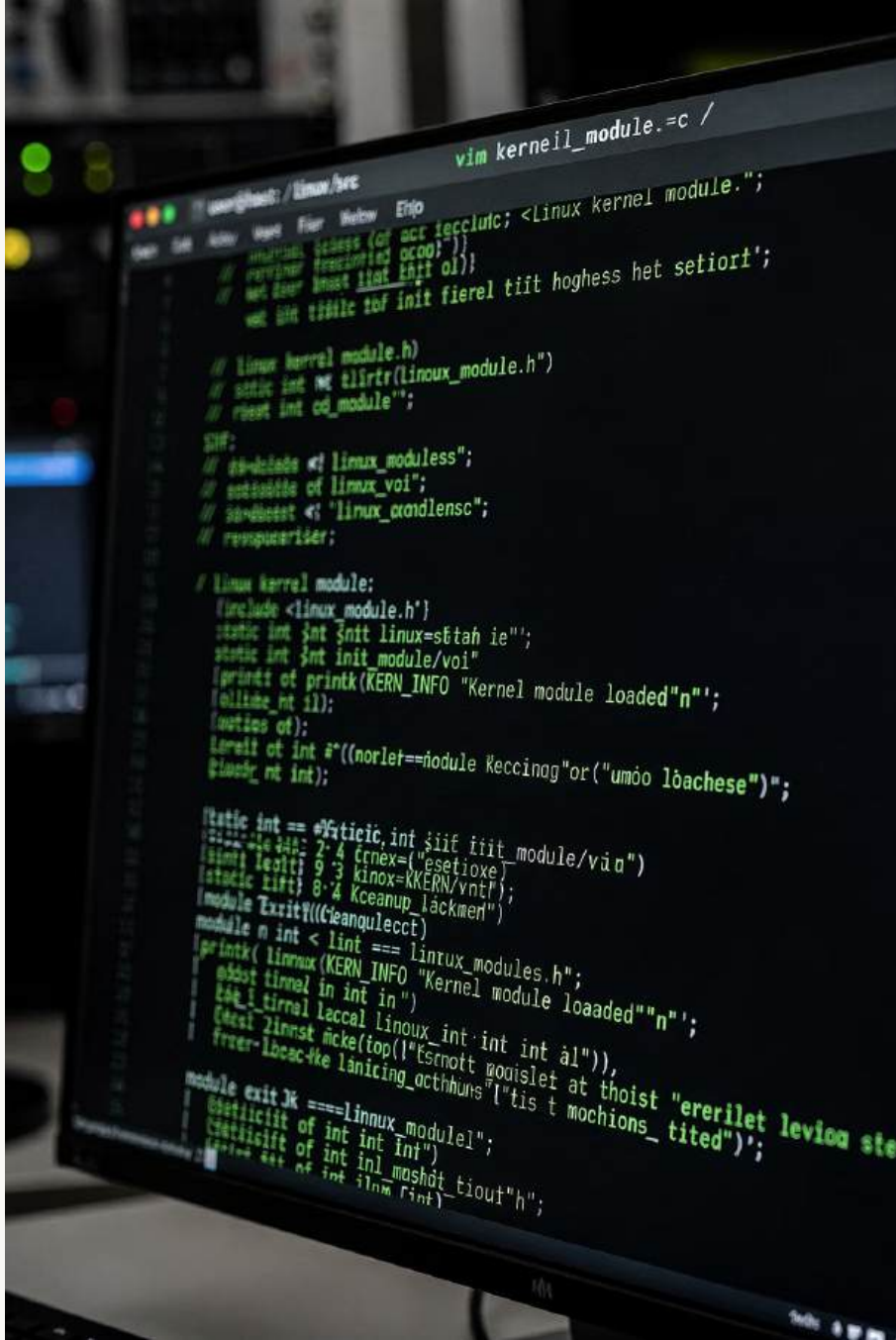
Кредити ЄКТС

Відповідно до стандарту

4

Модулі

Змістовних блоки курсу



Загальна інформація про дисципліну

Ідентифікація

Назва: Системне програмне забезпечення

Спеціальність: 123 Комп'ютерна інженерія

Ступінь: Бакалавр

Обсяг: 120 годин / 4 кредити ЄКТС

Модулів: 4 змістовних блоки

Призначення дисципліни

Дисципліна «Системне програмне забезпечення» формує у студентів системне розуміння принципів побудови та функціонування системного програмного забезпечення: операційних систем, компіляторів, інтерпретаторів, завантажувачів, редакторів зв'язків та утиліт системного рівня. Курс охоплює архітектуру операційних систем, управління процесами, пам'яттю, файловими системами, а також методи системного програмування в середовищах Linux та Windows.

Дисципліна є обов'язковою для підготовки бакалаврів зі спеціальності «Комп'ютерна інженерія» та забезпечує формування фахових компетентностей, передбачених освітньо-професійною програмою.

Загальні компетентності

Аналітичне мислення

Здатність до аналізу, синтезу та критичного оцінювання технічної інформації: виявлення причинно-наслідкових зв'язків у роботі системного ПЗ, порівняння архітектурних рішень, формування обґрунтованих висновків.

Комунікація

Здатність до усної та письмової комунікації: чітко описувати архітектурні рішення, аргументовано захищати лабораторні роботи, оформлювати технічні звіти та аналітичні записки.

Самоорганізація

Уміння самостійно навчатися, планувати роботу та організовувати виконання завдань у визначені строки, зокрема під час підготовки лабораторних, самостійних та індивідуальних завдань.

Командна робота

Відповідальність, ініціативність та готовність працювати в команді під час групового аналізу архітектурних кейсів, обговорення технічних рішень та спільного розв'язання практичних задач.

Студент повинен вміти: аналізувати технічну та наукову інформацію; виділяти головне та узагальнювати результати; планувати власну діяльність і дотримуватися дедлайнів; коректно спілкуватися з викладачами та одногрупниками; брати участь у командній роботі та розподіляти обов'язки.

Фахові компетентності

Архітектура системного ПЗ

Розуміння структури та принципів функціонування системного програмного забезпечення: операційних систем, компіляторів, завантажувачів, редакторів зв'язків. Знання основних архітектур ОС (монолітна, мікроядерна, гібридна) та їх порівняльний аналіз.

Управління ресурсами ОС

Здатність аналізувати механізми управління процесами, потоками, пам'яттю та введенням-виведенням в сучасних операційних системах. Знання алгоритмів планування процесів і стратегій управління пам'яттю.

Системне програмування

Знання принципів системного програмування: розробка системних викликів, робота з файловими системами, мережевим стеком, синхронізація потоків. Вміння розробляти системні програми з використанням API операційних систем Linux та Windows.

Інструменти та засоби

Орієнтування у сучасних засобах системного програмування: компілятори GCC/LLVM, відлагоджувачі, профілери, інструменти аналізу продуктивності. Використання наукових та технічних ресурсів для розроблення системного ПЗ.

Результати навчання

01

Пояснення ключових понять

Пояснювати сутність ключових понять: процес, потік, планування, синхронізація, дедлок, системний виклик, файлова система, сегментація, сторінкова організація пам'яті, компілятор, інтерпретатор.

02

Характеристика механізмів ОС

Характеризувати основні механізми сучасних операційних систем, застосовувати методи порівняльного аналізу алгоритмів планування та стратегій управління пам'яттю, визначати переваги та недоліки різних підходів.

03

Розробка системних програм

Розробляти системні програми з використанням API операційних систем Linux та Windows: реалізовувати управління процесами, файлові операції, міжпроцесову взаємодію та синхронізацію потоків.

04

Робота з технічною літературою

Орієнтуватися у науковій та технічній базі з системного програмування; використовувати сучасні інформаційні ресурси та документацію для підготовки обґрунтованих технічних рішень.

МОДУЛЬ 1

Архітектура операційних систем та управління процесами

1

Тема 1.1

Вступ до системного програмного забезпечення. Класифікація та архітектура операційних систем

2

Тема 1.2

Процеси та потоки в операційних системах. Стани процесів, PCB, дескриптори потоків

3

Тема 1.3

Планування процесів та потоків. Алгоритми планування: FCFS, SJF, Round Robin, пріоритетне

4

Тема 1.4

Синхронізація процесів та потоків. Критична секція, семафори, м'ютекси, монітори. Дедлоки

МОДУЛЬ 1 · ТЕМА 1.1

Вступ до СПЗ. Архітектура операційних систем

Лекційний матеріал (2 год.)

Лекція 1: Поняття системного програмного забезпечення. Класифікація СПЗ: операційні системи, транслятори, редактори зв'язків, завантажувачі, утиліти. Функції та компоненти ОС. Ядро ОС: монолітне, мікроядерне, гібридне, екзоядро. Системні виклики як інтерфейс між прикладним ПЗ та ОС.

Огляд сучасних ОС: Windows NT-архітектура, Linux-ядро (версії 5.x–6.x), macOS/Darwin. Порівняльний аналіз архітектурних рішень. Завантаження ОС: BIOS/UEFI, завантажувач GRUB2, ініціалізація ядра. Поняття API та ABI.

Ключові питання лекції

- Класифікація та призначення системного програмного забезпечення
- Типи архітектур ОС та їх порівняльний аналіз
- Механізм системних викликів та привілейовані режими
- Процес завантаження сучасних операційних систем
- Поняття API/ABI та їх роль у системному програмуванні

Практичне заняття (2 год.)

Завдання 1. Вивчення структури Linux-ядра: ознайомлення з деревом каталогів вихідного коду ядра, основні підсистеми (mm, fs, net, drivers). Складання схеми архітектури ядра Linux.

Завдання 2. Кейс: порівняльний аналіз монолітної та мікроядерної архітектур на прикладах Linux (монолітне) та QNX/MINIX (мікроядерне). Обговорення переваг і недоліків.

Завдання 3. Тестування за ключовими термінами теми (10 питань).

Самостійна робота (7 год.)

- Опрацювання підручника: Панченко В. І. та ін. «Системне програмне забезпечення. Програмування системних механізмів ОС» — розділ «Архітектура ОС»
- Підготовка реферату: «Еволюція архітектури операційних систем: від монолітних до мікроядерних»
- Складання порівняльної таблиці архітектур сучасних ОС

Очікуваний результат: студент вміє характеризувати типи архітектур ОС, пояснювати механізм системних викликів та описувати процес завантаження ОС.

МОДУЛЬ 1 · ТЕМА 1.2

Процеси та потоки в операційних системах

1

Створення

fork()/exec() у Linux. CreateProcess() у Windows. PCB, PID, таблиця процесів.

2

Стани процесу

Новий, готовий, виконується, заблокований, завершений. Діаграма переходів станів.

3

Потоки

Потоки (threads): POSIX Threads (pthreads). Легковагові та важковагові процеси.

4

Завершення

exit(), wait(), waitpid(). Зомбі-процеси. Сигнали SIGCHLD, SIGKILL, SIGTERM.

Лекційний матеріал (2 год.)

Поняття процесу та потоку. Блок керування процесом (PCB). Стани процесів та діаграма переходів. Системні виклики для роботи з процесами у Linux: fork(), exec(), exit(), wait(). Потоки: відмінність від процесів, POSIX Threads. Моделі потоків: M:1, 1:1, M:N. Контекст переключення. Міжпроцесова взаємодія (IPC): pipes, FIFO, черги повідомлень, розподілена пам'ять, сокети.

Практика (2 год.) та СРС (7 год.)

Практика: Розробка програм на C для створення та управління процесами у Linux (fork, exec, wait); демонстрація переходів між станами процесів; аналіз виводу команд ps, top, /proc.

СРС: Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення. Програмування системних механізмів ОС» (НТУ «ХПІ», 2024) — розділ «Процеси та потоки»; підготовка практичного завдання з IPC-механізмами.

МОДУЛЬ 1 · ТЕМИ 1.3–1.4

Планування процесів та синхронізація

Тема 1.3 — Планування процесів (Лекція 3, 2 год.)

Критерії планування: завантаженість CPU, пропускна здатність, час відгуку, час очікування. Алгоритми планування: FCFS (First-Come, First-Served), SJF (Shortest Job First), SRTF, Round Robin, пріоритетне планування, багаторівневі черги. Планування потоків. Порівняльний аналіз алгоритмів за критеріями ефективності. Планування в Linux (CFS — Completely Fair Scheduler) та Windows.

Практика (2 год.): Моделювання алгоритмів планування: розрахунок середнього часу очікування та відгуку для FCFS, SJF, RR з різними квантами часу; аналіз CFS у Linux через /proc/sched_debug.

СРС (7 год.): Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення» (2024) — розділ «Планування»; підготовка порівняльної таблиці алгоритмів планування за ключовими характеристиками.

Тема 1.4 — Синхронізація та дедлоки (Лекція 4, 2 год.)

Проблема критичної секції. Умови гонки (race conditions). Засоби синхронізації: м'ютекси (mutex), семафори (Dijkstra), монітори, умовні змінні. Класичні задачі синхронізації: «Виробник-Споживач», «Читачі-Письменники», «Обідні філософи». Дедлоки: умови виникнення (взаємне виключення, утримання та очікування, відсутність витіснення, кругове очікування). Методи обробки дедлоків: запобігання, уникнення (алгоритм банкіра), виявлення та відновлення.

Практика (2 год.): Реалізація синхронізації потоків за допомогою pthreads (mutex, semaphore, condition variables); демонстрація race condition та її усунення; аналіз дедлоків.

СРС (7 год.): Опрацювання: Мосіюк О. О. «Операційні системи та системне програмування» (КПІ, 2022) — розділ «Синхронізація»; розробка програми з ілюстрацією задачі «Виробник-Споживач».

МОДУЛЬ 2

Управління пам'яттю та файловими системами

Тема 2.1

Управління пам'яттю: логічний та фізичний адресний простір, сегментація, сторінкова організація, TLB.

Тема 2.2

Віртуальна пам'ять. Сторінкова заміна: алгоритми OPT, FIFO, LRU, Clock. Фрагментація.

Тема 2.3

Файлові системи: структура, типи (FAT32, NTFS, ext4, Btrfs). Операції з файлами та каталогами.

Тема 2.4

Підсистема введення-виведення. Драйвери пристроїв. Буферизація. DMA. Алгоритми дискового планування.

МОДУЛЬ 2 · ТЕМА 2.1

Управління пам'яттю в операційних системах

Лекційний матеріал (2 год.)

Логічний та фізичний адресний простір. Прив'язка адрес: на етапі компіляції, завантаження, виконання. Методи розподілу пам'яті: фіксовані розділи, динамічні розділи (first fit, best fit, worst fit). Сегментація: сегментна таблиця, переваги та недоліки. Сторінкова організація пам'яті: таблиця сторінок, TLB (Translation Lookaside Buffer), багаторівневі таблиці сторінок. Комбінована сегментно-сторінкова організація. Спільна пам'ять (shared memory) у Linux та Windows.

Ключові питання

- Відмінність логічного та фізичного адресного простору
- Принцип роботи TLB та його роль у продуктивності
- Порівняння сегментації та сторінкової організації
- Механізм спільної пам'яті між процесами
- Фрагментація пам'яті: зовнішня та внутрішня

Практичне заняття (2 год.)

Завдання 1. Аналіз карти пам'яті процесу у Linux (/proc/PID/maps): визначення сегментів коду, даних, стеку, купи, динамічних бібліотек. Побудова схеми адресного простору.

Завдання 2. Кейс: порівняння алгоритмів пошуку вільного блоку (first fit, best fit, worst fit) за показниками фрагментації та швидкості виділення. Розрахункові задачі.

Завдання 3. Тестування за ключовими термінами теми (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення. Програмування системних механізмів ОС» (НТУ «ХПІ», 2024) — розділ «Управління пам'яттю»
- Підготовка реферату: «Механізми управління пам'яттю в Linux-ядрі: SLUB-алокатор та buddy-система»
- Розробка програми для роботи зі спільною пам'яттю (POSIX shared memory)

Очікуваний результат: студент вміє описувати механізми управління пам'яттю, пояснювати роботу TLB та розраховувати характеристики різних алгоритмів розподілу пам'яті.

МОДУЛЬ 2 · ТЕМА 2.2

Віртуальна пам'ять та алгоритми сторінкової заміни

Концепція

Принцип локальності. Робочий набір (Working Set). Сторінкові переривання (page faults).

FIFO / LRU

FIFO — найстаріша сторінка. LRU — найдавніше використана. Апаратна підтримка LRU.

Linux / Windows

Реалізація в Linux (LRU-списки, kswapd). Windows Working Set Manager.

OPT

Оптимальний алгоритм. Теоретичний мінімум замін. Недоступний на практиці — еталон порівняння.

Clock / NFU

Алгоритм «Годинник» — наближення до LRU. NFU (Not Frequently Used). Thrashing.

Лекційний матеріал (2 год.)

Поняття віртуальної пам'яті. Принцип локальності (просторова та часова). Сторінкові переривання (page fault): обробка, стратегії заповнення. Алгоритми заміни сторінок: OPT, FIFO, LRU, Clock (Second Chance), NFU. Поняття «пробуксовки» (thrashing): причини та методи запобігання. Модель робочого набору. Аномалія Белейді. Реалізація у Linux та Windows.

Практика (2 год.) та СРС (7 год.)

Практика: Моделювання алгоритмів сторінкової заміни: розрахунок кількості сторінкових переривань для заданої рядкового числа сторінок та кількості фреймів для FIFO, LRU, Clock. Демонстрація аномалії Белейді на прикладі FIFO.

СРС: Опрацювання: Мосіюк О. О. «Операційні системи та системне програмування» (КПІ, 2022) — розділ «Віртуальна пам'ять»; підготовка аналітичної записки «Реалізація управління пам'яттю у ядрі Linux 6.x».

МОДУЛЬ 2 · ТЕМИ 2.3–2.4

Файлові системи та підсистема введення–виведення

Тема 2.3 — Файлові системи (Лекція 7, 2 год.)

Поняття файлу та файлової системи. Структура файлової системи: суперблок, inode, блоки даних, каталоги. Типи файлових систем: FAT32, NTFS, ext4, Btrfs, XFS, ZFS. Операції з файлами та каталогами через системні виклики (open, read, write, close, stat). VFS (Virtual File System) у Linux — абстракція над реальними ФС. Права доступу та атрибути файлів. Журналювання (journaling).

Практика (2 год.): Робота з файловою системою через системні виклики C: відкриття, зчитування, запис, позиціонування (lseek). Аналіз структури inode через stat(). Перегляд монтованих ФС.

СРС (7 год.): Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення» (2024); підготовка реферату «Порівняльний аналіз файлових систем ext4 та Btrfs для серверних середовищ».

Тема 2.4 — Підсистема введення–виведення (Лекція 8, 2 год.)

Архітектура підсистеми В/В. Класифікація пристроїв: блокові та символьні. Концепція драйвера пристрою. Рівні В/В: програмне опитування (polling), переривання, DMA. Буферизація та кешування. Диспетчери В/В. Алгоритми планування дискових запитів: FCFS, SSTF, SCAN, C-SCAN, LOOK. Порівняльний аналіз. Підсистема В/В у Linux: sysfs, udev. Порти та переривання.

Практика (2 год.): Аналіз підсистеми В/В у Linux: команди lsblk, blkid, iostat; моделювання алгоритмів дискового планування за заданою чергою запитів; розрахунок загального переміщення головки.

СРС (7 год.): Підготовка аналітичної записки «Алгоритми планування дискових запитів: порівняльний аналіз та застосування в SSD-накопичувачах».

МОДУЛЬ 3

Транслятори та системи програмування

Тема 3.1

Структура та класифікація трансляторів. Компілятори та інтерпретатори. Фази компіляції.

Тема 3.2

Лексичний та синтаксичний аналіз. Граматики та мови. Скінченні автомати. Парсери.

Тема 3.3

Семантичний аналіз та генерація проміжного коду. Оптимізація коду. Таблиця символів.

Тема 3.4

Генерація машинного коду. Редактори зв'язків. Завантажувачі. Динамічні бібліотеки (DLL/SO).

МОДУЛЬ 3 · ТЕМА 3.1

Структура та класифікація трансляторів

Лекційний матеріал (2 год.)

Поняття транслятора. Класифікація: компілятори (ahead-of-time), інтерпретатори, JIT-компілятори (Just-In-Time), транспілятори. Фази компіляції: лексичний аналіз → синтаксичний аналіз → семантичний аналіз → генерація проміжного коду → оптимізація → генерація цільового коду. Front-end та back-end компілятора. Архітектура GCC та LLVM/Clang. Переваги компіляції та інтерпретації: порівняльний аналіз. Байт-код та віртуальні машини (JVM, CLR).

Ключові питання

- Відмінності між компілятором, інтерпретатором та JIT-компілятором
- Фази компіляції та їх роль у трансформації коду
- Архітектура GCC та LLVM: front-end і back-end
- Поняття проміжного представлення (IR) коду
- Байт-код та принцип роботи віртуальних машин

Практичне заняття (2 год.)

Завдання 1. Огляд фаз компіляції на прикладі GCC: генерація препроцесованого коду, асемблерного коду, об'єктного файлу та виконуваного файлу (gcc -E, -S, -c, -o). Аналіз вихідних файлів кожної фази.

Завдання 2. Кейс: порівняльний аналіз продуктивності компільованих (C/C++) та інтерпретованих (Python) програм для однакових обчислювальних задач. Висновки щодо застосування.

Завдання 3. Тестування за ключовими термінами теми (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Цибульник С. О., Барандич К. С. «Технології розроблення програмного забезпечення» (КПІ, 2022) — розділ «Транслятори»
- Підготовка реферату: «Архітектура LLVM/Clang: модульний підхід до побудови компіляторів»
- Аналіз асемблерного виводу GCC для типових конструкцій мови C

Очікуваний результат: студент вміє характеризувати типи трансляторів, описувати фази компіляції та пояснювати архітектуру сучасних компіляторних інфраструктур.

МОДУЛЬ 3 · ТЕМА 3.2

Лексичний та синтаксичний аналіз

Лекційний матеріал (2 год.)

Лексичний аналіз (сканування): токени, лексеми, шаблони. Регулярні граматики та регулярні вирази. Скінченні детерміновані (DFA) та недетерміновані автомати (NFA). Побудова DFA за регулярним виразом (Thompson's construction, subset construction). Генератори сканерів: Lex/Flex. Синтаксичний аналіз (парсинг): контекстно-вільні граматики (CFG), дерево розбору (parse tree), абстрактне синтаксичне дерево (AST). Алгоритми парсингу: спадний (LL, рекурсивний спуск), висхідний (LR, SLR, LALR). Генератори парсерів: Yacc/Bison.

Практика (2 год.) та СРС (7 год.)

Практика: Побудова DFA для простого лексичного аналізатора; реалізація рекурсивного спуску для арифметичних виразів; візуалізація AST для фрагментів коду на C. Використання онлайн-інструментів (ANTLR Lab, Regex101).

СРС: Опрацювання: Цибульник С. О., Барандич К. С. «Технології розроблення програмного забезпечення» (КПІ, 2022) — розділ «Синтаксичний аналіз»; підготовка аналітичної записки «Порівняння LL та LR-парсерів: переваги, недоліки та сфери застосування».

Очікуваний результат: студент вміє будувати DFA для лексичних аналізаторів, пояснювати алгоритми парсингу та аналізувати AST реальних програм.

МОДУЛЬ 3 · ТЕМА 3.3

Семантичний аналіз та оптимізація коду

Семантичний аналіз

Перевірка типів (type checking). Таблиця символів: структура, операції. Статична та динамічна типізація. Область видимості (scope).

Проміжне представлення

Three-Address Code (TAC). SSA (Static Single Assignment). LLVM IR. DAG (Directed Acyclic Graph) для виразів.

Оптимізація коду

Машинно-незалежна: згортання констант, усунення мертвого коду, підстановка вбудованих функцій. Машинно-залежна: розподіл регістрів, вибір команд, планування інструкцій.

Лекційний матеріал (2 год.)

Семантичний аналіз: атрибутні граматики, перевірка типів, таблиця символів. Генерація проміжного коду: Three-Address Code, SSA-форма, LLVM IR. Оптимізації: локальні (у базових блоках), глобальні (між блоками), міжпроцедурні. Конкретні оптимізації: constant folding, dead code elimination, loop unrolling, inlining, CSE (Common Subexpression Elimination). Рівні оптимізації компілятора GCC: -O0, -O1, -O2, -O3, -Os.

Практика (2 год.) та СРС (7 год.)

Практика: Аналіз впливу рівнів оптимізації GCC (-O0 vs -O2 vs -O3) на розмір та продуктивність згенерованого коду; дослідження LLVM IR за допомогою clang -emit-llvm; побудова таблиці символів для фрагменту С-програми.

СРС: Опрацювання: Трофименко О. Г., Манаков С. Ю., Ларін Д. Г. «Основи програмної інженерії» (Одеса: Фенікс, 2022) — розділ «Компілятори та оптимізація»; підготовка есе «LLVM як платформа для побудови компіляторів нового покоління».

МОДУЛЬ 3 · ТЕМА 3.4

Генерація коду. Компонувальники та завантажувачі

Лекційний матеріал (2 год.)

Генерація цільового коду: вибір команд (instruction selection), розподіл регістрів (register allocation), планування інструкцій. Формат об'єктних файлів: ELF (Executable and Linkable Format) у Linux, PE (Portable Executable) у Windows. Структура ELF: заголовок, секції (.text, .data, .bss, .rodata). Редактор зв'язків (linker): статичне та динамічне компонування. Символи та таблиця символів. Бібліотеки: статичні (.a, .lib) та динамічні (.so, .dll). Завантажувач (loader): процес завантаження програми, переміщення адрес (relocation). Dynamic linker (ld.so у Linux).

Ключові питання

- Структура ELF-файлу та його секції
- Відмінності статичного та динамічного компонування
- Роль завантажувача в підготовці програми до виконання
- Динамічне зв'язування та DLL/SO бібліотеки
- Переміщення адрес (relocation) та його механізм

Практичне заняття (2 год.)

Завдання 1. Аналіз структури ELF-файлів за допомогою утиліт readelf та objdump: перегляд заголовка, секцій, таблиці символів. Порівняння статично та динамічно скомпонованих виконуваних файлів.

Завдання 2. Розробка та компіляція власної динамічної бібліотеки (.so) на C: реалізація функцій, компіляція з -fPIC, завантаження через dlopen/dlsym. Демонстрація механізму динамічного зв'язування.

Завдання 3. Тестування за ключовими термінами теми (10 питань).

Самостійна робота (7 год.)

- Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення» (НТУ «ХПІ», 2024) — розділ «Компонувальники та завантажувачі»
- Підготовка реферату: «ELF vs PE: порівняння форматів виконуваних файлів Linux та Windows»
- Складання схеми процесу компіляції, компонування та завантаження програми

Очікуваний результат: студент вміє аналізувати структуру ELF-файлів, пояснювати процес статичного та динамічного компонування, розробляти та використовувати динамічні бібліотеки.

МОДУЛЬ 4

Системне програмування та безпека ОС

Тема 4.1

Системне програмування в Linux: POSIX API, робота з файлами, процесами, сигналами, міжпроцесова взаємодія.

Тема 4.2

Системне програмування в Windows: Win32 API, потоки, синхронізація, реєстр, файлова система NTFS.

Тема 4.3

Безпека операційних систем. Моделі захисту. Автентифікація, авторизація, аудит. Уразливості ОС.

Тема 4.4

Сучасні тенденції: контейнеризація (Docker, namespaces, cgroups), хмарні ОС, вбудовані системи.

МОДУЛЬ 4 · ТЕМА 4.1

Системне програмування в Linux (POSIX API)

Лекційний матеріал (2 год.)

Стандарт POSIX та його роль у переносимості системних програм. Робота з файлами: open(), read(), write(), lseek(), close(), fcntl(). Маніпуляції з процесами: fork(), exec(), wait(), getpid(), getppid(). Сигнали: signal(), sigaction(), kill(), raise(). Таймери: alarm(), setitimer(). Механізми IPC: анонімні та іменовані канали (pipes, FIFO), черги повідомлень POSIX, семафори POSIX, розподілена пам'ять POSIX (shm_open, mmap). UNIX-сокети. Системні виклики ptrace та strace для трасування.

Ключові питання

- Стандарт POSIX і його значення для системного програмування
- Низькорівнева робота з файлами через файлові дескриптори
- Механізми IPC: переваги та недоліки різних підходів
- Обробка сигналів у системних програмах
- Трасування системних викликів за допомогою strace

Практичне заняття (4 год.)

Завдання 1. Розробка системної програми: реалізація командного інтерпретатора (мінішелу) з підтримкою виконання команд, перенаправлення stdin/stdout, передачі через pipe між командами. Використання fork(), exec(), dup2(), pipe().

Завдання 2. Реалізація клієнт-серверного додатку з використанням UNIX-сокетів та обробкою сигналів (SIGCHLD для зомбі-процесів). Демонстрація IPC через спільну пам'ять.

Завдання 3. Аналіз системних викликів реальних програм за допомогою strace; порівняння кількості та типів системних викликів для різних програм.

Самостійна робота (7 год.)

- Опрацювання: Панченко В. І. та ін. «Системне програмне забезпечення. Програмування системних механізмів ОС» (НТУ «ХПІ», 2024) — розділи «POSIX API», «IPC»
- Підготовка реферату «Розробка багатопроцесних програм на POSIX: кращі практики та типові помилки»
- Розробка власного демон-процесу (daemon) з коректним завершенням роботи

Очікуваний результат: студент вміє розробляти системні програми з використанням POSIX API, реалізовувати IPC-механізми та обробляти сигнали в Linux.

МОДУЛЬ 4 · ТЕМА 4.2

Системне програмування в Windows (Win32 API)

Лекційний матеріал (2 год.)

Архітектура Windows NT: рівень HAL, ядро, виконавча система (executive), підсистеми. Win32 API: принципи роботи, HANDLE, об'єкти ядра. Управління процесами та потоками: CreateProcess(), CreateThread(), TerminateProcess(). Синхронізація: Events, Mutexes, Semaphores, Critical Section, WaitForSingleObject(), WaitForMultipleObjects(). Робота з файлами: CreateFile(), ReadFile(), WriteFile(), GetFileAttributes(). Реєстр Windows: RegOpenKey(), RegQueryValue(). Сповіщення файлової системи: ReadDirectoryChangesW(). Динамічні бібліотеки (DLL): LoadLibrary(), GetProcAddress(), FreeLibrary().

Практика (2 год.) та СРС (7 год.)

Практика: Розробка програм на C із використанням Win32 API: багатопотоковий додаток із синхронізацією через Mutex та Event; моніторинг змін файлової системи через ReadDirectoryChangesW(); завантаження та виклик функцій з DLL через LoadLibrary/GetProcAddress. Аналіз у Process Monitor (Sysinternals).

СРС: Опрацювання: матеріали з курсу (НТУ «ХПІ», 2024) — тема «Програмування системних механізмів ОС Windows»; підготовка аналітичної записки «Порівняння механізмів синхронізації в Linux (pthreads) та Windows (Win32 API)»; складання порівняльної таблиці системних викликів Linux та Windows API.

Очікуваний результат: студент вміє розробляти системні програми з використанням Win32 API, реалізовувати багатопотоковість та синхронізацію в Windows.

МОДУЛЬ 4 · ТЕМИ 4.3–4.4

Безпека ОС та сучасні тенденції системного ПЗ

Тема 4.3 — Безпека операційних систем (Лекція 15, 2 год.)

Моделі захисту: дискреційний (DAC), мандатний (MAC), рольовий (RBAC) контроль доступу. Автентифікація, авторизація, аудит (AAA). Механізми безпеки Linux: права доступу UNIX, ACL, SELinux, AppArmor, capabilities. Механізми безпеки Windows: ACL, UAC, Windows Defender Credential Guard, TPM. Типові уразливості системного ПЗ: переповнення буфера (buffer overflow), use-after-free, race conditions. Механізми захисту: ASLR, DEP/NX, stack canaries, PIE. Інструменти аналізу: Valgrind, AddressSanitizer.

Практика (2 год.): Демонстрація та аналіз вразливості переповнення буфера; огляд механізмів ASLR та stack canaries; налаштування прав доступу за допомогою ACL у Linux; аналіз SELinux-політик.

СРС (7 год.): Опрацювання: Задерейко О. В. та ін. (Одеса: Фенікс, 2022) — розділ «Безпека ОС»; підготовка реферату «Механізми захисту від атак на переповнення буфера в сучасних ОС».

Тема 4.4 — Сучасні тенденції системного ПЗ (Лекція 16, 2 год.)

Контейнеризація: технології namespaces та cgroups у Linux як основа Docker. Принцип роботи контейнерів, відмінність від віртуальних машин. Оркестрація: Kubernetes — базові концепції. Мікроядра та unikernel: ОС для хмарних середовищ. Вбудовані операційні системи (RTOS): FreeRTOS, Zephyr — особливості планування реального часу. ОС для IoT. Мова Rust для системного програмування: безпека пам'яті без GC. WebAssembly (WASM) як нова парадигма системного ПЗ. Гіпервізори (Type 1 та Type 2): KVM, Xen, VMware.

Практика (2 год.): Практична робота з Docker: створення контейнера, Dockerfile, namespaces та cgroups у дії (docker stats, /proc). Огляд можливостей мови Rust для системного програмування. Обговорення перспектив розвитку системного ПЗ.

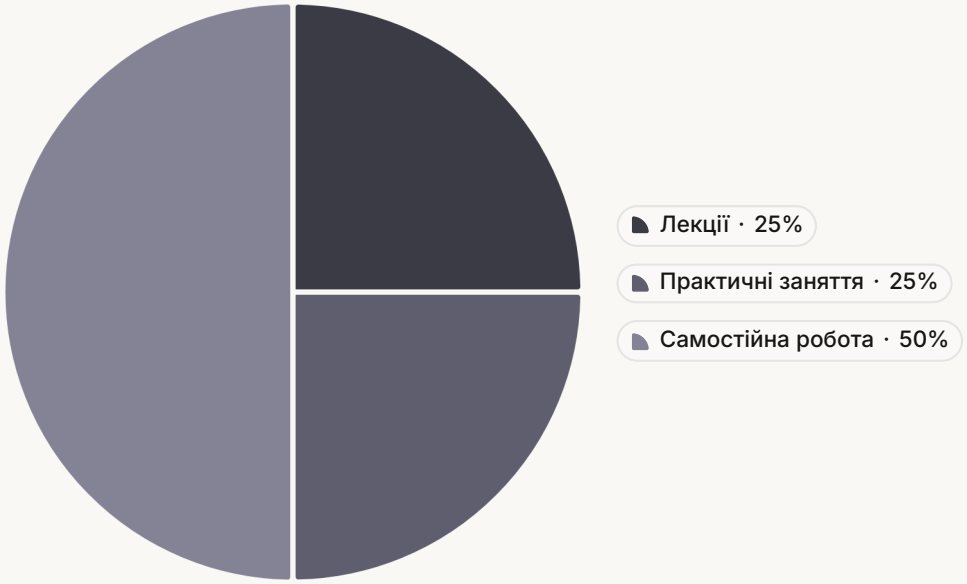
СРС (7 год.): Опрацювання: Корочкін О. В., Кулаков Ю. О., Русанова О. В. «Програмне забезпечення високопродуктивних комп'ютерних систем» (КПІ, 2024) — розділ «Сучасні платформи»; підготовка аналітичної записки «Контейнеризація vs. Віртуалізація: порівняння підходів для хмарних середовищ».

Розподіл навчального часу — 120 годин

Модуль	Тема	Лекції (год.)	Практичні (год.)	СРС (год.)
Модуль 1	Тема 1.1 — Архітектура ОС	2	2	7
	Тема 1.2 — Процеси та потоки	2	2	7
	Тема 1.3 — Планування процесів	2	2	7
	Тема 1.4 — Синхронізація та дедлоки	2	2	7
Модуль 2	Тема 2.1 — Управління пам'яттю	2	2	7
	Тема 2.2 — Віртуальна пам'ять	2	2	7
	Тема 2.3 — Файлові системи	2	2	7
	Тема 2.4 — Підсистема введення-виведення	2	2	7
Модуль 3	Тема 3.1 — Транслятори: класифікація	2	2	7
	Тема 3.2 — Лексичний та синтаксичний аналіз	2	2	7
	Тема 3.3 — Семантика та оптимізація коду	2	2	7
	Тема 3.4 — Компонувальники та завантажувачі	2	2	7
Модуль 4	Тема 4.1 — Системне програмування Linux	2	4	7
	Тема 4.2 — Системне програмування Windows	2	2	7
	Тема 4.3 — Безпека операційних систем	2	2	7
	Тема 4.4 — Сучасні тенденції СПЗ	2	2	7
Разом	**16 тем**	**30**	**30**	**60**

Загальний обсяг: 120 годин = 4 кредити ЄКТС. Розподіл є орієнтовним і може коригуватися відповідно до специфіки навчального процесу.

Структура навчального часу



Коментар до розподілу

Загальний обсяг — **120 годин (4 кредити ЄКТС)** — розподілено між трьома формами роботи для забезпечення балансу між теорією та практикою.

Лекції — 30 год. (25%)

Системне подання ключового теоретичного матеріалу з архітектури ОС, управління ресурсами, теорії трансляторів та системного програмування з використанням схем, діаграм і прикладів коду.

Практичні — 30 год. (25%)

Розробка системних програм, аналіз механізмів ОС, моделювання алгоритмів, розгляд кейсів та захист практичних завдань. Рівна частка з лекціями підкреслює практичну спрямованість курсу.

Самостійна робота — 60 год. (50%)

Поглиблене вивчення, підготовка завдань, робота з технічною документацією та науковою літературою. Самостійна робота становить **50% обсягу дисципліни**.

Самостійна робота студентів — Детальний опис

Опрацювання літератури

Поглиблене вивчення теоретичного матеріалу за підручниками та посібниками з системного програмного забезпечення (Панченко, Мосіюк, Цибульник, Корочкін та ін.), технічною документацією (man-pages, POSIX, MSDN). Студент конспектує ключові положення, порівнює підходи різних авторів.

Індивідуальні завдання

Виконання технічних завдань (аналіз алгоритмів, моделювання механізмів ОС, порівняльний аналіз підходів); розробка системних програм; підготовка рефератів, есе та аналітичних записок з актуальних питань системного програмування.

Робота з інформаційними ресурсами

Опрацювання офіційних ресурсів: kernel.org (документація Linux-ядра), docs.microsoft.com (Win32 API), man7.org (Linux man-pages), POSIX стандарт (IEEE Std 1003.1), репозиторії KPI та НТУ «ХПІ». Аналіз вихідного коду ядра Linux на github.com/torvalds/linux.

Підготовка до занять

Підготовка до практичних занять (повторення теорії, написання коду); підготовка до тестувань (ключові поняття, алгоритми, системні виклики); підготовка до контрольних робіт за кожним модулем (систематизація знань, типові задачі).

Форми контролю знань

1	2	3
Поточний контроль	Проміжний контроль	Підсумковий контроль
Завдання: тестові вправи та технічні задачі за темами змістовного модуля. Оцінюються регулярність виконання завдань, якість програмного коду та рівень засвоєння матеріалу. Поточний контроль формує накопичувальну оцінку протягом семестру. Проводиться після кожної теми у формі тестування (10–15 питань) або захисту практичної роботи.	Комплексне практичне завдання або реферат, що охоплюють теми змістовних модулів (теоретичні питання, розробка програм, аналіз механізмів ОС). Оцінюються повнота відповіді, коректність коду та обґрунтованість висновків. Проводиться після завершення кожного змістовного модуля.	Іспит відповідно до навчального плану. Форма проведення — тестування, письмові відповіді на теоретичні питання та практичне завдання з системного програмування. Підсумкова оцінка враховує результати поточного, проміжного та підсумкового контролю: **100-бальна система** з переведенням у національну шкалу та ЄКТС (A, B, C, D, E, FX, F).

Методи викладання

Викладання дисципліни «Системне програмне забезпечення» поєднує класичні академічні та практично орієнтовані методи, що забезпечує формування міцної теоретичної бази та практичних навичок системного програмування, необхідних для майбутньої інженерної діяльності у сфері комп'ютерної інженерії.

Лекції

Системне подання теоретичного матеріалу з використанням схем архітектур, діаграм станів, таблиць порівняння алгоритмів та фрагментів коду; студенти ведуть конспекти та задають уточнювальні запитання.

Робота з документацією

Опрацювання технічної документації (man-pages, POSIX, MSDN, kernel.org); формування навичок самостійного пошуку технічної інформації та критичного аналізу документації.

Кейс-стаді

Аналіз реальних технічних рішень і архітектур (Linux-ядро, GCC, Docker); студенти виявляють інженерні компроміси та обґрунтовують вибір на основі технічних критеріїв.

Технічні дискусії

Обговорення інженерних рішень та архітектурних компромісів у системному програмуванні; розвиток критичного мислення, технічної аргументації та поваги до альтернативних підходів.

Практичне програмування

Розробка системних програм із використанням POSIX API та Win32 API; налагодження за допомогою gdb, strace, Valgrind, WinDbg; формування навичок низькорівневого програмування.

Моделювання алгоритмів

Розрахункові задачі та моделювання алгоритмів планування, сторінкової заміни, дискового планування; порівняльний аналіз ефективності за ключовими метриками.

Усі методи спрямовані на формування компетентностей, передбачених освітньо-професійною програмою спеціальності 123 «Комп'ютерна інженерія».

Рекомендована література — Підручники та посібники (2022–2024 рр.)

1. Панченко В. І., Коломійцев О. В., Межерицький С. Г.
Системне програмне забезпечення. Програмування системних механізмів ОС : навч. посібник. — Харків : НТУ «ХПІ», 2024. — 327 с.

2. Мосіюк О. О.
Операційні системи та системне програмування : навч.-метод. посібник. — Київ : КПІ ім. Ігоря Сікорського, 2022.

3. Цибульник С. О., Барандич К. С.
Технології розроблення програмного забезпечення : підручник. Ч. 1 : Життєвий цикл програмного забезпечення. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 270 с.

4. Корочкін О. В., Кулаков Ю. О., Русанова О. В.
Програмне забезпечення високопродуктивних комп'ютерних систем : конспект лекцій. — Київ : КПІ ім. Ігоря Сікорського, 2024.

5. Трофименко О. Г., Манаков С. Ю., Ларін Д. Г.
Основи програмної інженерії : навч.-метод. посібник. — Одеса : Фенікс, 2022.

6. Кублій Л. І.
Алгоритми та структури даних. Основи алгоритмізації : підручник. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 528 с.

7. Висоцька В. А., Оборська О. В.
Python: алгоритмізація та програмування : навч. посібник. — Львів : Новий Світ-2000, 2023. — 514 с.

8. Добровольський Ю. Г.
Стандартизація в інженерії програмного забезпечення : навч. посібник. — Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича, 2022. — 140 с.

Додаткова література та наукові статті

9. Рудий Т. В., Паранчук Я. С., Сеник В. В.

Алгоритмізація та програмування. Ч. 1 : Структурне програмування : навч. посібник. — Львів : Львівський держ. ун-т внутр. справ, 2023. — 240 с. *Рекомендується для тем 1.1–1.2 (алгоритмічна база).*

10. Тищенко К. В., Логвинов А. М., Пилипенко О. В.

Алгоритмічні мови програмування в електронних інформаційних системах та комп'ютерних технологіях (практикум) : навч. посібник. — Суми : СумДУ, 2024. — 95 с. *Рекомендується для практичних занять модулів 3–4.*

11. Баран С. В.

Розробка програмного забезпечення з використанням патернів проектування : навч. посібник. — Кривий Ріг : Державний університет економіки і технологій, 2023. — 203 с. *Рекомендується для тем 4.1–4.2 (системне програмування).*

12. Марченко О. І., Вітковська І. І.

Компоненти програмної інженерії. Ч. 1 : Вступ до програмної інженерії: лабораторний практикум. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 50 с. *Рекомендується для модулів 3–4.*

Нормативно-правова база

- Конституція України

  Прийнята 28 червня 1996 р. Основний Закон держави. **zakon.rada.gov.ua**

- Закон «Про освіту» від 05.09.2017 № 2145-VIII

  Визначає засади освітньої діяльності в Україні. **zakon.rada.gov.ua**

- Закон «Про вищу освіту» від 01.07.2014 № 1556-VII

  Регулює відносини у сфері вищої освіти: структура, ступені, кредити ЄКТС, стандарти. **zakon.rada.gov.ua**

- Стандарт вищої освіти України: спеціальність 123 «Комп'ютерна інженерія»

  Визначає компетентності та результати навчання для підготовки бакалаврів. **mon.gov.ua**

- POSIX стандарт (IEEE Std 1003.1-2017)

  Міжнародний стандарт інтерфейсів операційних систем. Основа курсу з системного програмування. **pubs.opengroup.org**

Офіційні ресурси та фахові видання

The Linux Kernel Archives

kernel.org — офіційний архів вихідного коду ядра Linux, документація, патчі, стабільні версії.

Linux man-pages

man7.org — повна документація системних викликів та бібліотечних функцій POSIX Linux.

Microsoft Learn (Win32 API)

learn.microsoft.com — офіційна документація Microsoft: Win32 API, Windows Driver Kit, системне програмування Windows.

Електронний репозиторій КПІ

ela.kpi.ua — навчально-методичні матеріали КПІ ім. Ігоря Сікорського з системного програмування та ОС.

Репозиторій НТУ «ХПІ»

repository.kpi.kharkov.ua — наукові публікації та навчальні посібники з системного програмного забезпечення.

Національна бібліотека України ім. В. І. Вернадського

nbuv.gov.ua — електронні ресурси, наукові статті, монографії з комп'ютерних наук.

Міністерство освіти і науки України

mon.gov.ua — освітні стандарти, навчальні програми, методичні матеріали зі спеціальності 123.

GitHub — Linux Kernel

github.com/torvalds/linux — дзеркало вихідного коду ядра Linux для вивчення системних механізмів на реальних прикладах.

Зв'язок дисципліни зі спеціальністю 123 «Комп'ютерна інженерія»

Чому «Системне програмне забезпечення» для інженерів?

Дисципліна «Системне програмне забезпечення» є ключовою складовою підготовки бакалаврів зі спеціальності 123 «Комп'ютерна інженерія». Розуміння принципів роботи операційних систем та системного програмного забезпечення є фундаментом для проектування ефективних комп'ютерних систем будь-якого рівня.

Сучасний інженер у сфері комп'ютерної інженерії постійно взаємодіє з системним ПЗ — від вибудови embedded-систем до розгортання хмарних платформ. Глибоке розуміння механізмів ОС є конкурентною перевагою на ринку праці.

Практичне значення для майбутньої кар'єри

- Розробка системного та вбудованого ПЗ для комп'ютерних систем (embedded, RTOS)
- Проектування та оптимізація продуктивності серверних та хмарних застосунків
- Розробка драйверів пристроїв та низькорівневих системних компонентів
- Розуміння механізмів безпеки ОС для захисту комп'ютерних систем
- Контейнеризація та DevOps: Docker, Kubernetes, CI/CD пайплайни
- Навички налагодження та профілювання системних програм (gdb, Valgrind, perf)

Ключові компетентності курсу — Підсумок

Курс охоплює всі ключові аспекти системного програмного забезпечення — від архітектури операційних систем до сучасних технологій контейнеризації — і формує у студентів цілісне розуміння принципів побудови та функціонування системного ПЗ сучасних комп'ютерних систем.

Бажаємо успіхів у вивченні Системного програмного забезпечення!

120

Годин

Загальний обсяг дисципліни

4

Кредити ЄКТС

Відповідно до стандарту

16

Тем

У 4 змістовних модулях

8

Джерел

Рекомендованої літератури 2022–2024 рр.

> «Той, хто розуміє, як працює система зсередини, ніколи не буде обмежений її поверхнею.»

Спеціальність 123 · Комп'ютерна інженерія · Бакалавр · 4 кредити ЄКТС