СИЛАБУС

Кафедра автоматизованого управління технологічними процесами

Назва курсу	Системне програмне забезпечення
Мова викладання	Українська
Викладач (-і)	Нікітенко Є.В., доцент кафедри комп'ютерних та інформаційних технологій
Профайл викладача	E-mail: evnikitenko@gmail.com ResearcherID: **R-4441-2016** ORCID: http://orcid.org/0000-0002-9222-644X
Контакти викладача	E-mail: evnikitenko@gmail.com

1. Анотація курсу

Навчальна дисципліна «Системне програмне забезпечення» є складовою циклу професійної підготовки фахівців освітньо-кваліфікаційного рівня «бакалавр».

Курс підпорядковано розробці ефективної конфігурації вбудованої системи Лінукс для створення середовища для запуску прикладних програм; застосування принципів розробки модулів ядра Лінукс для організації взаємодії з апаратними засобами системи.

Короткий зміст курсу:

Тема 1: Архітектура системи Linux. Технічні вимоги. Linux і операційна система Unix. Три стандартні канали input/output. Зручне комбінування інструментів. Модульна конструкція, призначена для використання іншими користувачами. Псевдокод. Архітектура системи Linux. Application Binary Interface. Доступ до вмісту реєстру через вбудовану збірку. Доступ до вмісту регістра управління через вбудовану збірку. Рівні привілеїв ЦП. Рівні привілеїв або кільця на x86. Архітектура Linux. Бібліотеки. Системні виклики. Контексти виконання в ядрі. Контекст процесу. Контекст переривання.

Тема 2: Віртуальна пам'ять. Технічні вимоги. Віртуальна пам'ять. Рішення 1 - спрощений помилковий підхід. Непряме звернення. Переклад адрес. Переваги використання VM. Ізоляція процесу. Захист області пам'яті. Тестування програми memcpy(). Структура пам'яті процесу. Сегменти або зіставлення. Текстовый сегмент. Сегменти даних. Сегменти бібліотеки. Сегмент стека.

Тема 3: Обмеження ресурсів. Ліміти ресурсів. Деталізація лімітів ресурсів. Типи ресурсів. Доступні ліміти ресурсів. Жорсткі і м'які обмеження.

Запит і зміна значень ліміту ресурсів. Застереження. Утиліта prlimit. Використання prlimit(1). API-інтерфейси. Приклади коду.

Тема 4: Розподіл динамічної пам'яті. Сімейство API glibc malloc (3). API-інтерфейс malloc (3). malloc (3) - деякі поширені запитання. malloc (3) - короткий опис. Calloc API. Realloc API. API-інтерфейс realloc array. Використання sbrk() API. Як насправді поводиться malloc (3). Приклад коду - malloc(3). Резидент. Блокування пам'яті. Ліміти і привілеї. Блокування всіх сторінок. Захист пам'яті. Захист пам'яті - приклад коду. Журнали LSM, Ftrace. Експеримент: запуск програми memprot на ARM-32. Ключі захисту пам'яті. Використання alloca для автоматичного виділення пам'яті.

Тема 5: Проблеми з пам'яттю в Linux. Загальні проблеми з пам'яттю. Неправильний доступ до пам'яті. Доступ і/або використання неініціалізованих змінних. Тестовий приклад 1: не ініціалізований доступ до пам'яті. Доступ до пам'яті поза межами. Тестовий набір. Тестові приклади. Помилки використання після звільнення/використання після повернення. Витік. Невизначена поведінка. Фрагментація.

Тема 6: Інструменти налагодження для проблем з пам'яттю. Типи інструментів. Valgrind. Використання інструменту Memcheck від Valgrind. Зведена таблиця Valgrind. Valgrind за і проти: короткий опис. Збірка програм для використання з ASan. Запуск тестових прикладів з ASan. Зведена таблиця AddressSanitizer (ASan). AddressSanitizer за і проти. Параметри Malloc. Покриття коду при тестуванні. Що робити сучасному розробнику C/C++?

Тема 7. Облікові дані процесу. Традиційна модель дозволів Unix. Дозволи на рівні користувача. Як працює модель дозволів Unix. Визначення категорії доступу. Біти спеціального дозволу setuid і setgid. Установка бітів setuid і setgid за допомогою chmod. Спроба злому 1. Системні виклики. Запит облікових даних процесу. Приклад коду. sudo - як це працює. Установка облікових даних процесу. Спроба злому 2. Скрипт для визначення встановлених програм setuid-root і setgid. setgid приклад. Відмова від привілеїв. Збережено набір UID - швидка демонстрація. Системні виклики setres [u|g]id(2). Важливі зауваження з безпеки.

Тема 8: Можливості процесу. Сучасна модель можливостей POSIX. Мотивація. Можливості POSIX. Можливості - деякі важливі подробиці. Підтримка ОС. Перегляд можливостей процесу через procfs. Набори можливостей файлів. Вбудовування можливостей в бінарний код. Можливості dumb виконавчі файли. Getcap і схожі утиліти. Wireshark. Програмна установка можливостей. Як ls відображає різні виконавчі файли. Рівні дозволів. Поради з безпеки. Під капотом на рівні ядра.

Тема 9: Виконання процесу. Технічні вимоги. Перетворення програми в процес. Аксіома exec Unix. Ключові моменти під час виконання операції. Перевірка аксіоми exec. Експеримент 1 - на CLI спрощений. Експеримент 2 - на CLI. Точка неповернення. API сімейства exec. Обробка помилок і exec. Передача нуля як аргумент. Вказівка імені спадкоємця. Решта API сімейства exec. Execle API. APIінтерфейс execv. Exec на рівні ОС. Зведена таблиця - сімейство API-інтерфейсів exec. Приклад коду.

Тема 10: Створення процесу. Як працює fork. Використання системного виклику fork. Правило fork # 1. Правило fork # 2. Правило fork # 3. Атомне виконання. Правило fork # 4 - дані. Правило fork # 5 - гонки. Процес і відкриті файли. Правило fork # 6 - відкриття файлів. Відкриті файли і безпеку. malloc і fork. Коротко про COW. Simple shell project. Семантика fork-exec в Unix. Необхідність wait. Виконання wait. Перемога в гонці після fork. API wait - подробиці. Сценарії wait. Варіанти очікування - API. wait (2). waitid (2). Фактичний системний виклик. vfork. Зомбі. Правило fork # 7. Правила fork - резюме.

Тема 11: Сигналізація - Частина I. Чому сигнали? Коротко про сигнальному механізмі. Доступні сигнали. Стандартні або Unixсигнали. Обробка сигналів. Використання системного виклику sigaction для перехоплення сигналів. Макроси тестування функцій. Маскують сигнали. Маскування сигналів за допомогою sigprocmask API. Запит маски сигналу. Обробка сигналів в ОС - опитування без переривань. Reentrant-функції. Функції, захищені від асинхронних сигналів. Альтернативні способи забезпечити обробник сигналів. Сигнально-безпечні атомарні числа. Flags sigaction. Зомбі. Класичний спосіб і сучасний спосіб боротьби із зомбі. Flag SA_NOCLDSTOP. Перервані системні виклики і способи їх усунення за допомогою SA_RESTART. Flag SA_RESETHAND. Робота з SA_NODEFER. Випадок 1: біт SA_NODEFER скинутий. Випадок 2: біт SA_NODEFER встановлений. Виконання випадку 1 - біт SA_NODEFER скинутий [за замовчуванням]. Виконання випадку 2 - біт SA_NODEFER встановлений. Використання альтернативного стека сигналів. Реалізація для обробки сигналів великого обсягу за допомогою альтернативного стека сигналів. Випадок 1 - дуже маленький (100 КБ) стек альтернативних сигналів. Випадок 2: великий (16 МБ) стек альтернативних сигналів. Різні підходи до обробки сигналів.

Тема 12: Сигналізація - Частина II. Витончена обробка збоїв процесу. Деталізація інформації за допомогою SA_SIGINFO. Структура siginfo_t.Отримання відомостей на рівні системи при збої процесу. Захоплення і вилучення інформації з аварії. Регістр скидання. Пошук місця збою в вихідному коді. Сигналізація - застереження і підводні камені. Витончене поводження з errno. Виправлення помилки errno race. Правильний sleep. Системний виклик nanosleep. Сигнали в реальному часі. Відмінності від стандартних сигналів. Сигнали в реальному часі і пріоритет. Відправлення сигналів. Kill. Сигналізація як IPC. LTTng. Альтернативні методи обробки сигналів. Синхронне очікування сигналів. Пауза. Очікування вічно або поки не надійде сигнал. Синхронне блокування сигналів через sigwait * API. Бібліотека sigwait API. Системні виклики sigwaitinfo і sigtimedwait. Signalfd (2) API.

Тема 13: Таймери. Старі інтерфейси. Будильник. Alarm API. Таймери інтервальні. Прості цифровий годинник CLI. Отримання поточного часу. Пробні запуски. Новий механізм POSIX (інтервального) таймера. Типовий робочий процес додатки. Створення і використання POSIX (інтервального)

таймера. Гонка - вмикання і вимикання таймера POSIX. Запит таймера. Приклад фрагмента коду, що показує робочий процес. Інтервальні таймери POSIX - приклади програм. React - пробні запуски. Гра React - перегляд коду. Додаток для таймер бігу: ходьби.

2. Мета та цілі курсу

Метою викладання навчальної дисципліни «Системне програмне забезпечення» є необхідність формування у студентів знань примітивів виконання на синхронізації в ядрі Лінукс, деталі їх реалізації, вплив на швидкодію системи; алгоритмів функціонування різних типів модулів пристроїв в ядрі Лінукс, способи їх використання для організації взаємодії з системним програмним забезпеченням; особливостей взаємодії з апаратними засобами спеціалізованих систем.

Значення дисципліни для реалізації вимог кваліфікаційної характеристики фахівця та вивчення наступних дисциплін полягає в тому, що дисципліна сприяє формуванню алгоритмічного мислення майбутнього фахівця, створює базу, яка необхідна при вивченні багатьох наступних дисциплін. Виходячи з цього викладання дисципліни «Системне програмне забезпечення» підпорядковане вирішенню таких основних задач, як з'ясування концептуальних принципів проєктування ОС Linux та її вбудовані та серверні програми, що є критично важливими компонентами сьогоднішньої ключової програмної інфраструктури в децентралізованому та мережевому всесвіті.

3. Результати навчання

Здатність до абстрактного мислення, аналізу та синтезу.

Здатність застосовувати знання у практичних ситуаціях.

Знання та розуміння предметної області та розуміння професійної діяльності.

Здатність до пошуку, оброблення та аналізу інформації з різних джерел.

Здатність генерувати нові ідеї (креативність).

Здатність працювати в команді.

Здатність до математичного та логічного мислення, формулювання та досліджування математичних моделей, зокрема дискретних математичних моделей, обґрунтовування вибору методів і підходів для розв'язування теоретичних і прикладних задач у галузі комп'ютерних наук, аналізу та інтерпретування.

Здатність до побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проєктування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем.

Здатність опанувати сучасні методи математичного моделювання об’єктів, процесів і явищ, розробляти моделі й алгоритми чисельного розв’язування задач математичного моделювання з урахуванням похибок наближеного чисельного розв’язування професійних задач.

Здатність здійснювати формалізований опис задач дослідження операцій в організаційно-технічних і соціально-економічних системах різного призначення, визначати їх оптимальні рішення, будувати моделі оптимального вибору управління з урахуванням змін параметрів економічної ситуації, оптимізувати процеси управління в системах різного призначення та рівня ієрархії.

4. Обсяг курсу

Вид заняття	Загальна к-сть годин
лекції	16
лабораторні заняття	14
Практичні заняття	14
самостійна робота (реферат, РГР, КР, КП, тощо)	76

5. Пререквізити

Дисципліна «Системне програмне забезпечення» є базовою для отримання більш глибоких знань про механізми явищ, що виникають. В значній мірі це стосується тих систем, структурна та динамічна складність яких робить неефективним чи взагалі неможливим використання аналітичних методів дослідження.

В результаті вивчення дисципліни «Системне програмне забезпечення» студенти не тільки краще засвоюють теоретичні та практичні знання, але й оволодівають навичками використання пакетів прикладних програм.

6. Система оцінювання та вимоги

Загальна система оцінювання курсу	Підсумкова оцінка з дисципліни є сумою оцінок з відповідною вагою за кожен з таких видів робіт: активна робота на лабораторних та практичних заняттях, тести та підсумковий контроль (екзамен). Підсумкова оцінка визначається відповідно до поданої нижче таблиці оцінювання за різними шкалами (100-бальна, ECTS, національна).
Розрахункова графічна-робота	В рамках курсу не передбачено виконання РГР.
Лабораторні та практичні роботи	**Критерії оцінювання лабораторних та практичних робіт:** 1. Підготовленість до

	лабораторних/практичних занять 2. Самостійність виконання лабораторних/практичних робіт. 3. Повнота виконання завдань 4. Своєчасність виконання та захисту лабораторних/практичних робіт Максимальний бал за кожну лабораторну/практичну роботу – 5 балів
Тест	Проміжний тест проводиться у кожному модулі з курсу та оцінюється максимально в 10 балів.
Іспит	Іспит проводиться в кінці курсу, включає два теоретичних питання і одне практичне завдання. Максимально оцінюється в 40 балів.
Умови допуску до підсумкового контролю	Позитивна оцінка за всіма обов'язковими видами робіт (лабораторні та практичні роботи)

7. Політики курсу

Політика щодо академічної доброчесності: списування під час тесту, іспиту заборонені.

Жодні форми порушення академічної доброчесності не толеруються. У випадку таких подій – реагування відповідно до Методичних рекомендацій для закладів вищої освіти з підтримки принципів академічної доброчесності.

Правила перезарахування кредитів у випадку мобільності, правила перескладання або відпрацювання пропущених занять тощо: відбувається згідно з Положення про організацію освітнього процесу у Таврійському національному університет ім. В.І. Вернадського.

Політика щодо дедлайнів та перескладання: роботи, які здаються із порушенням термінів без поважних причин, оцінюються на нижчу оцінку (до -50% від можливої максимальної кількості балів за вид діяльності).

Перескладання тесту відбувається за наявності поважних причин (наприклад, лікарняний).

Політика щодо відвідування: відвідування занять є обов'язковим компонентом. За об'єктивних причин (наприклад, хвороба, працевлаштування, міжнародне стажування) навчання може відбуватись в онлайн-формі за погодженням.

8. Рекомендована література

Базова література

1. Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. Linux Device Drivers, Third Edition - 2005, O'Reilly Media, Inc. - ISBN: 0-596-00590-3

2. Ulrich Drepper. What Every Programmer Should Know About Memory - 2007 - https://people.freebsd.org/~lstewart/articles/cpumemory.pdf
3. Paul E. McKenney. Is Parallel Programming Hard, And, If So, What Can You Do About It? - Linux Technology Center, 2017 - https://mirrors.edge.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html
4. Andrew Sloss, Dominic Symes, Chris Wright. ARM System Developer's Guide: Designing and Optimizing System Software. - Morgan Kaufmann; 2004 - ISBN-13: 978-1558608740

Додаткові

1. Mastering Linux Kernel Development by Raghu Bharadwaj.
2. The Art of Linux Kernel Design Illustrating the Operating System Design Principle and Implementation by Lixiang Yang.
3. Hands On: C Programming and Unix Application Design: UNIX System Calls and Subroutines using C c A. D. Marshall 1998-2004.
4. Practical Linux Security Cookbook - Second Edition. Tajinder Kalsi ISBN: 978-1-78913-839-9.
5. Hands-On Linux Administration on Azure Frederik Vos ISBN: 978-1-78913-096-6