

KARTA KURSU (realizowanego w specjalności)**Inżynieria oprogramowania**

(nazwa specjalności)

| | |
|-----------------|--------------------------------|
| Nazwa | Programowanie systemowe |
| Nazwa w j. ang. | System programming |

| | | |
|-----------------|---|----------------------|
| Koordinator | dr Wojciech Gwizdała | Zespół dydaktyczny |
| | | dr Wojciech Gwizdała |
| Punktacja ECTS* | st. stacjonarne: 4 st. niestacjonarne: 4 | |

Opis kursu (cele kształcenia)

Celem kursu jest przekazanie studentom wiedzy oraz umiejętności niezbędnych do programowania na poziomie systemowym, ze szczególnym uwzględnieniem środowiska Unix/Linux. Studenci zapoznają się z mechanizmami działania systemu operacyjnego oraz uczą się tworzyć programy współpracujące bezpośrednio z systemem — w tym zarządzanie pamięcią, obsługa procesów i wątków, komunikacja międzyprocesowa oraz operacje wejścia/wyjścia na niskim poziomie. Kurs kładzie nacisk na rozumienie struktury systemów operacyjnych, wykorzystanie interfejsów API systemowych oraz stosowanie narzędzi typowych dla programistów systemowych. Rozwijane są również umiejętności pisania bezpiecznego i wydajnego kodu podczas pracy nad zadaniami typowymi dla inżyniera oprogramowania. Kurs prowadzony jest w języku polskim.

Warunki wstępne

| | |
|--------------|---|
| Wiedza | Student posiada wiedzę z zakresu podstaw programowania w językach wysokiego poziomu (np. C, C++). Zna podstawowe koncepcje systemów operacyjnych (procesy, pamięć, system plików). Rozumie ogólne zasady budowy systemów informatycznych oraz ich komponentów. |
| Umiejętności | Student potrafi pisać i uruchamiać proste programy użytkowe w wybranym języku programowania. Potrafi korzystać z narzędzi programistycznych, takich jak edytory (vi, vim, nano) oraz kompilatorów (gcc, g++). Umie pracować z linią komend w systemach uniksopodobnych. |
| Kursy | Programowanie, Programowanie obiektowe, Organizacja i architektura komputerów, Systemy operacyjne |

Efekty uczenia się

| | Efekt uczenia się dla kursu | Odniesienie do efektów kierunkowych |
|--------|---|-------------------------------------|
| Wiedza | Po zakończeniu kursu student: | |
| | W01: Zna mechanizmy działania systemów operacyjnych: procesy, wątki, pamięć, system plików, I/O | S1_W01 |
| | W02: Posiada zaawansowaną wiedzę na temat programowania systemowego | S1_W02 |
| | W03: Rozumie wpływ zarządzania zasobami systemowymi na jakość i niezawodność oprogramowania | S1_W03 |

| | Efekt uczenia się dla kursu | Odniesienie do efektów kierunkowych |
|--------------|---|-------------------------------------|
| Umiejętności | Po zakończeniu kursu student: | |
| | U01: Potrafi pisać programy w językach C/C++ wykorzystujące API systemowe (np. POSIX). | S1_U01, S1_U02 |
| | U02: Umie debugować, testować oraz kontrolować jakość kodu w aplikacjach systemowych. | S1_U03 |
| | U03: Potrafi tworzyć programy operujące na plikach, procesach, sygnałach, pamięci współdzielonej. | S1_U04, S1_U06 |

| | Efekt uczenia się dla kursu | Odniesienie do efektów kierunkowych |
|-----------------------|--|-------------------------------------|
| Kompetencje społeczne | Po zakończeniu kursu student: | |
| | K01: Potrafi samodzielnie uzupełniać wiedzę z zakresu programowania systemowego. | S1_K01, S1_K04 |
| | K02: Potrafi formułować krytyczne opinie na temat stosowanych rozwiązań systemowych. | S1_K02 |

Studia stacjonarne

| Organizacja | | | | | | | |
|---------------|------------|---------------------|---|----|---|---|---|
| Forma zajęć | Wykład (W) | Ćwiczenia w grupach | | | | | |
| | | A | K | L | S | P | E |
| Liczba godzin | 15 | | | 15 | | | 1 |

Studia niestacjonarne

| Organizacja | | | | | | | |
|---------------|------------|---------------------|---|----|---|---|---|
| Forma zajęć | Wykład (W) | Ćwiczenia w grupach | | | | | |
| | | A | K | L | S | P | E |
| Liczba godzin | 10 | | | 10 | | | 1 |

Opis metod prowadzenia zajęć

Zajęcia prowadzone są w formie wykładów oraz laboratoriów, z naciskiem na praktyczne zastosowanie wiedzy w środowisku systemowym (Unix/Linux). Wykorzystywane metody dydaktyczne to:

- praca laboratoryjna – samodzielna i zespołowa implementacja programów wykorzystujących funkcje systemowe (procesy, wątki, pamięć, I/O);
- ćwiczenia kodowe na żywo – wspólne analizowanie i modyfikowanie kodu systemowego w środowisku terminalowym;
- wprowadzenie problemowe – prezentacja kluczowych koncepcji (np. komunikacja międzyprocesowa, deskryptory plików, zarządzanie zasobami);
- analiza przypadków błędów i wycieków – wykorzystywanie narzędzi diagnostycznych (gdb, valgrind, strace);
- konsultacje techniczne i wsparcie prowadzącego – pomoc w debugowaniu, optymalizacji i strukturze kodu.

Zajęcia mają na celu rozwijanie nie tylko umiejętności technicznych, ale również odpowiedzialnego podejścia do programowania niskopoziomowego, bezpieczeństwa oraz jakości kodu.

Formy sprawdzania efektów uczenia się

| | E – learning | Gry dydaktyczne | Ćwiczenia w szkole | Zajęcia terenowe | Praca laboratoryjna | Projekt indywidualny | Projekt grupowy | Udział w dyskusji | Referat | Praca pisemna (esej) | Egzamin ustny | Egzamin pisemny | Zadania problemowe |
|-----|--------------|-----------------|--------------------|------------------|---------------------|----------------------|-----------------|-------------------|---------|----------------------|---------------|-----------------|--------------------|
| W01 | | | | | X | X | | X | | | | X | |
| W02 | | | | | X | X | | X | | | | X | |
| W03 | | | | | X | X | | X | | | | X | |
| U01 | | | | | X | X | | | | | | X | |
| U02 | | | | | X | X | | | | | | X | |
| U03 | | | | | X | X | | | | | | X | |
| K01 | | | | | X | X | | | | | | | |
| K02 | | | | | X | X | | | | | | | |

| | |
|----------------|---|
| Kryteria oceny | <p>Zaliczenie laboratorium</p> <p>Wykonie projektu oraz aktywne uczestnictwo w zajęciach jest podstawą zaliczenia laboratorium. Zaliczenie tej części zajęć jest warunkiem koniecznym dopuszczenia do egzaminu. Ocena z laboratorium nie jest wystawiana – przedmiot kończy się oceną z egzaminu.</p> |
| | <p>Egzamin końcowy</p> <p>Egzamin obejmuje treści wykładów i ma formę testu teoretyczno-praktycznego. Sprawdza wiedzę dotyczącą działania jądra systemu operacyjnego, znajomości technik i narzędzi projektowania, programowania i testowania aplikacji systemowych.</p> <p>Ocena końcowa</p> <p>Ocena końcowa pochodzi wyłącznie z egzaminu końcowego.</p> <p>Zaliczenie na ocenę dostateczną otrzymuje student, który potrafi:</p> <ul style="list-style-type: none"> • wyjaśnić podstawowe mechanizmy działania systemu operacyjnego (procesy, pamięć, pliki); • napisać prosty program systemowy z użyciem podstawowych wywołań systemowych (np. tworzenie procesu, operacje na plikach); • poprawnie skompilować, uruchomić i przetestować aplikację systemową w środowisku Unix/Linux; • wykonać prostą dokumentację techniczną przygotowanego rozwiązania. <p>Zaliczenie na ocenę dobrą lub bardzo dobrą otrzymuje student, który spełnia warunki oceny dostatecznej, a oprócz tego także:</p> <ul style="list-style-type: none"> • potrafi wykorzystać zaawansowane mechanizmy programowania systemowego (IPC, wątki, pamięć współdzielona); • stosuje dobre praktyki programistyczne (czytelność kodu, modularność, obsługa błędów); • korzysta z narzędzi do debugowania, profilowania oraz kontroli jakości kodu; • analizuje i optymalizuje działanie aplikacji systemowych; • potrafi krytycznie ocenić zastosowane rozwiązania projektowe. <p>Obecność na wykładach jest warunkiem koniecznym zaliczenia tej części kursu i dopuszczenia do egzaminu.</p> |

Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do programowania systemowego
 - Różnice między programowaniem użytkowym a systemowym
 - Przegląd interfejsów systemowych: POSIX
 - Kompilacja, linkowanie i struktura programu systemowego
2. Procesy i wątki
 - Tworzenie i zarządzanie procesami (fork(), exec(), wait())
 - Wątki (pthread, CreateThread) – podstawy i synchronizacja
 - Harmonogramowanie, stany procesu, identyfikatory
3. Pamięć i zarządzanie zasobami
 - Pamięć dynamiczna (malloc, free, mmap)
 - Segmentacja, stronicowanie, ochrona pamięci
 - Dostęp do pamięci współdzielonej
4. Operacje wejścia/wyjścia
 - Operacje na plikach: otwieranie, czytanie, zapisywanie (open, read, write)
 - Obsługa błędów, deskryptory plików
 - Strumień I/O, buforowanie
5. Komunikacja międzyprocesowa (IPC)
 - Potoki (pipes), kolejki komunikatów, semaforey
 - Sygnały i ich obsługa (signal, sigaction)
6. Narzędzia i dobre praktyki
 - Debugowanie (gdb, strace, valgrind)
 - Analiza i profilowanie wydajności

Wykaz literatury podstawowej

1. Kardaś M., STM32. Aplikacje i ćwiczenia w języku C z biblioteką HAL, Warszawa: Helion, 2020.
2. Dariusz Bismor, Programowanie systemów sterowania. Narzędzia i metody, Warszawa: Wydawnictwo Naukowe PWN, 2021.
3. Love R., Linux. Programowanie systemowe. Wydanie II, Gliwice: Helion, 2021.

Wykaz literatury uzupełniającej

1. Hyde R., Profesjonalne programowanie. Część 2. Myśl niskopoziomowo, pisz wysokopoziomowo, Gliwice: Helion, 2017.
2. Majdzik P., Programowanie współbieżne. Systemy czasu rzeczywistego, Warszawa: Wydawnictwo Naukowe PWN, 2014.
3. Kernighan B.W., Ritchie D.M., Język ANSI C, Warszawa: Wydawnictwo Naukowe PWN, 2018.

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) – **studia stacjonarne**

| | | |
|--|--|----|
| Liczba godzin w kontakcie z prowadzącymi | Wykład | 15 |
| | Konwersatorium (ćwiczenia, laboratorium itd.) | 15 |
| | Pozostałe godziny kontaktu studenta z prowadzącym | 15 |
| Liczba godzin pracy studenta bez kontaktu z prowadzącymi | Lektura w ramach przygotowania do zajęć | 10 |
| | Realizacja zadań domowych (problemowych) po zapoznaniu się z niezbędną literaturą przedmiotu | 15 |
| | Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna lub w grupie) | 10 |

| | | |
|--|---|-----|
| | Przygotowanie do egzaminu/zaliczenia | 20 |
| | Ogółem bilans czasu pracy | 100 |
| | Liczba punktów ECTS w zależności od przyjętego przelicznika | 4 |

Bilans godzinowy zgodny z CNPS (Całkowity Nakład Pracy Studenta) – **studia niestacjonarne**

| | | |
|---|--|-----|
| Liczba godzin w kontakcie z prowadzącymi | Wykład | 10 |
| | Konwersatorium (ćwiczenia, laboratorium itd.) | 10 |
| | Pozostałe godziny kontaktu studenta z prowadzącym | 15 |
| Liczba godzin pracy studenta bez kontaktu z prowadzącymi | Lektura w ramach przygotowania do zajęć | 20 |
| | Realizacja zadań domowych (problemowych) po zapoznaniu się z niezbędną literaturą przedmiotu | 15 |
| | Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna lub w grupie) | 20 |
| | Przygotowanie do egzaminu/zaliczenia | 20 |
| Ogółem bilans czasu pracy | | 100 |
| Liczba punktów ECTS w zależności od przyjętego przelicznika | | 4 |