

KARTA KURSU

Nazwa	Programowanie systemowe (C lub C++)
Nazwa w j. ang.	System programming (C or C++)

Koordynator	dr Wojciech Gwizdała	Zespół dydaktyczny
		dr Wojciech Gwizdała
Punktacja ECTS*	2	

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 używaniem terminala systemu operacyjnego oraz uczą się tworzyć programy współpracujące bezpośrednio z systemem — w tym zarządzanie systemem plików, obsługa procesów, operacje wejścia/wyjścia na niskim poziomie.

Kurs kładzie nacisk na wykorzystanie interfejsów API systemowych oraz stosowanie narzędzi typowych dla programistów systemowych. Rozwijane są również umiejętności pisania bezpiecznego i wydajnego kodu zapewniającego bezawaryjną pracę w środowisku systemu operacyjnego. Kurs prowadzony jest w języku polskim.

Warunki wstępne

Wiedza	Student posiada wiedzę z zakresu podstaw programowania w języku wysokiego poziomu (np. C/C++). Zna systemy liczbowe oraz jednostki informacji, a także podstawowe koncepcje i reguły stosowane w informatyce.
Umiejętności	Student potrafi pisać i uruchamiać proste programy użytkowe w języku C. Potrafi korzystać z narzędzi programistycznych, takich jak edytory oraz kompilatory.
Kursy	Programowanie, Teoretyczne podstawy informatyki

Efekty uczenia się

	Efekt uczenia się	Odniesienie do efektów kierunkowych
Wiedza	Po zakończeniu kursu student:	
	W01: Zna wybrane mechanizmy działania systemów operacyjnych: system plików, procesy, uprawnienia, operacje I/O	K_W02
	W02: Posiada wiedzę na temat programowania systemowego	K_W02, K_W03
	W03: Rozumie wpływ zarządzania zasobami systemowymi na jakość, niezawodność i bezpieczeństwo oprogramowania	K_W02, K_W03
Umiejętności	Efekt uczenia się	Odniesienie do efektów kierunkowych

	<p>Po zakończeniu kursu student:</p> <p>U01: Potrafi posługiwać się ważnymi aplikacjami systemowymi z poziomu terminala Unix/Linux</p> <p>U02: Umie pisać programy w językach C/C++ wykorzystujące biblioteki systemowe (np. POSIX)</p> <p>U03: Potrafi tworzyć, kompilować oraz testować programy wykonujące operacje I/O, działające na plikach i procesach</p> <p>U04: Umie dokonać analizy kodu w celu kontroli jego jakości i bezpieczeństwa w aplikacjach systemowych</p>	<p>K_U02</p> <p>K_U03</p> <p>K_U03, K_U04</p> <p>K_U03, K_U04</p>
Kompetencje społeczne	Efekt uczenia się	Odniesienie do efektów kierunkowych
	<p>Po zakończeniu kursu student:</p> <p>K01: Potrafi samodzielnie uzupełniać wiedzę z zakresu programowania systemowego</p> <p>K02: Potrafi formułować krytyczne opinie na temat stosowanych rozwiązań systemowych</p>	<p>K_K02</p> <p>K_K02</p>

Studia stacjonarne

Organizacja											
Forma zajęć	Wykład (W)	Ćwiczenia w grupach									
		A		K		L		S		P	E
Liczba godzin						25					

Studia niestacjonarne

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

Opis metod prowadzenia zajęć

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

- praca laboratoryjna – samodzielna implementacja programów wykorzystujących funkcje biblioteczne i systemowe (pliki, procesy, I/O);
- ćwiczenia kodowe na żywo – analizowanie i modyfikowanie kodu systemowego w środowisku terminalowym;
- 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 systemowego, jakości oraz bezpieczeństwa 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
W02					X	X		X					
W03					X	X		X					
U02					X	X							
U03					X	X							
U04					X	X							
K02					X	X		X					

Kryteria oceny	
Uwagi	Zaliczenie laboratorium Wykonie projektu, napisanie kolokwium oraz aktywne uczestnictwo w zajęciach jest podstawą zaliczenia laboratorium. Przedmiot kończy się zaliczeniem z oceną.
	Ocena końcowa Zaliczenie na ocenę dostateczną otrzymuje student, który potrafi: <ul style="list-style-type: none"> • wyjaśnić podstawowe pojęcia i mechanizmy działania systemu operacyjnego (pliki, procesy); • napisać prosty program systemowy z użyciem podstawowych wywołań systemowych (np. operacje na plikach, tworzenie procesu); • poprawnie skompilować, uruchomić i przetestować aplikację systemową w środowisku Unix/Linux;
	Zaliczenie na ocenę dobrą lub bardzo dobrą otrzymuje student, który spełnia warunki oceny dostatecznej, a oprócz tego także: <ul style="list-style-type: none"> • potrafi napisać bezpieczny program zapewniający obsługę błędów • stosuje dobre praktyki programistyczne (czytelność kodu i jego optymalizacja); • umie wykorzystać i omówić proste mechanizmy tworzenia oraz synchronizacji procesów; • analizuje i poprawia działanie aplikacji systemowych; • potrafi wykonać prostą dokumentację techniczną przygotowanego rozwiązania.

Treści merytoryczne (wykaz tematów)

1. Wprowadzenie do konsoli systemu operacyjnego Unix/Linux:
 - katalogi, pliki i linki;
 - uprawnienia;
 - procesy i sygnały;
 - potoki;
 - strumienie.
2. Automatyzacja zadań systemowych za pomocą skryptów w powłoce BASH.
3. Różnice pomiędzy programowaniem użytkowym a systemowym.
4. Biblioteka standardowa i funkcje systemowe Unix/Linux języka C.
5. Pisanie kodu, kompilacja i testowanie:
 - edytory vi, vim, nano,
 - kompilator gcc;
 - sposoby uruchamiania plików binarnych.
6. Programowanie wejścia/wyjścia:
 - operacje na plikach i katalogach: otwieranie, czytanie, zapisywanie, usuwanie;
 - deskryptory plików;
 - obsługa błędów;
5. Tworzenie procesów i zarządzanie nimi na poziomie kodu C:
 - funkcje fork() i exec();
 - sygnały i ich obsługa;
 - prosta synchronizacja procesów.
6. Odczytywanie i ustawianie czasu w środowisku Unix/Linux za pomocą funkcji systemowych.

Wykaz literatury podstawowej

1. Shotts W., Linux. Wprowadzenie do wiersza poleceń. Wydanie II, Gliwice: Helion, 2021.
2. Love R., Linux. Programowanie systemowe. Wydanie II, Gliwice: Helion, 2021.

Wykaz literatury uzupełniającej

1. Kernighan B.W., Ritchie D.M., Język ANSI C, Warszawa: Wydawnictwo Naukowe PWN, 2018.
2. Hyde R., Profesjonalne programowanie. Część 2. Myśl niskopoziomowo, pisz wysokopoziomowo, Gliwice: Helion, 2017.
3. Grębosz J., Opus magnum C++. Programowanie w języku C++. Wydanie III poprawione, Gliwice: Helion, 2024

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

Liczba godzin w kontakcie z prowadzącymi	Wykład	
	Konwersatorium (ćwiczenia, laboratorium itd.)	25
	Pozostałe godziny kontaktu studenta z prowadzącym	3
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	5
	Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna)	12
	Przygotowanie do egzaminu/zaliczenia	5
Ogółem bilans czasu pracy		50
Liczba punktów ECTS w zależności od przyjętego przelicznika		2

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

Liczba godzin w kontakcie z prowadzącymi	Wykład	
	Konwersatorium (ćwiczenia, laboratorium itd.)	15
	Pozostałe godziny kontaktu studenta z prowadzącym	5
Liczba godzin pracy studenta bez kontaktu z prowadzącymi	Lektura w ramach przygotowania do zajęć	10
	Przygotowanie projektu lub prezentacji na podany temat (praca indywidualna)	12
	Przygotowanie do egzaminu/zaliczenia	8
Ogółem bilans czasu pracy		50
Liczba punktów ECTS w zależności od przyjętego przelicznika		2