Projekt 2

Projekty należy wykonać w języku C. Interfejs programu powinien być zgodny z przykładową sesją. Dokumentacja projektu powinna zawierać również raport z testów. Wykonanie projektu obejmuje:

- plan aplikacji

- research  
- implementację  
- testowanie

- dokumentację  
- obronę projektu

- poprawę błędów

Deadline: 25.01.2024

Projekty należy wysłać na maila w spakowanym katalogu:

Nazwisko Imie

nazwa.c

nazwa.docx ; dokumentacja i raport z testów

W nazwach katalogu i plików nie używamy polskich znaków.

Projekt 1 (3.5)

Napisz program split, który podzieli plik podany jako pierwszy parametr na pliki o rozmiarze podanym jako drugi parametr. Można też używać rozmiarów predefiniowanych, np.: cd, dvd, blue-ray. Nazwy plików wynikowych powinny składać się z nazwy pliku wejściowego i odpowiedniego przyrostka, np.:

plik\_part1

plik\_part2

plik\_part3

Projekt 2 (4.0)

Napisz program line-end odnajdujący znak lub sekwencję znaków oznaczającą koniec linii. W dokumentacji opisz dokładnie algorytm odnajdywania końca linii. Program należy przygotować w jednym pliku źródłowym do skompilowania na systemie Linux i Windows. Przykładowa sesja:

System Windows

char newline: '\r' '\n'

hex newline: 0D 0A

Projekt 3 (4.5)

W programie jiffies zaimplementuj moduł jądra, który tworzy plik /proc/jiffies podający bieżącą wartość zmiennej jiffies, gdy plik ten jest czytany, na przykład za pomocą polecenia cat /proc/jiffies. Po usunięciu modułu należy usunąć utworzony plik.

Projekt 4 (4.5)

W programie seconds zaimplementuj moduł jądra, który tworzy plik /proc/sekundy podający liczbę sekund, które upłynęły od momentu załadowania modułu jądra, gdy plik ten jest czytany, na przykład za pomocą polecenia cat /proc/seconds. W programie należy użyć wartości jiffies oraz HZ. Po usunięciu modułu należy usunąć utworzony plik.

Projekt 5 (5.0)

Napisz w języku Java program Drzewo, który rekurencyjnie utworzy drzewo katalogowe zakodowane w tablicy trójwymiarowej. Można użyć komend tworzących plik i katalog oraz komend zmiany katalogu i przejścia do katalogu poziom wyżej. Zakoduj w tablicach trójwymiarowych poniższe drzewa katalogowe i przetestuj program.

\_\_\_A\_\_\_ \_\_X\_\_

/ / \ \ / | \

B C D f1 Y Z f1

/ \ \ /| |\

D f1 E X f1 Z f1

Sposób kodowania drzewa w tablicy i funkcja rekurencyjna tworząca drzewo muszą być dokładne opisane w dokumentacji.

Projekt 6 (5.0)

Napisz program add2path dla systemu Linux, który do zmiennej systemowej PATH w sposób permanentny:

- dodaje bieżący katalog, jeśli go tam nie ma

- usuwa bieżący katalog, jeśli on tam jest

- dodaje katalog przekazany jako parametr, jeśli go tam nie ma

- usuwa katalog przekazany jako parametr, jeśli on tam jest

- wypisuje na terminalu wykonaną operację

- katalogi mogą być przekazywane przez adresy względne i bezwzględne

Projekt 7 (5.0)

Napisz program add2path dla systemu Windows, który do zmiennej systemowej PATH w sposób permanentny:

- dodaje bieżący katalog, jeśli go tam nie ma

- usuwa bieżący katalog, jeśli on tam jest

- dodaje katalog przekazany jako parametr, jeśli go tam nie ma

- usuwa katalog przekazany jako parametr, jeśli on tam jest

- wypisuje na terminalu wykonaną operację

- katalogi mogą być przekazywane przez adresy względne i bezwzględne

Projekt 8 (5.0)

Napisz program check sprawdzający poprawność archiwum zip podanego jako parametr. Archiwum uznajemy za poprawne, jeśli zawiera katalog postaci Nazwisko Imie ze spacją w środku i bez polskich liter. Poprawność imienia należy zweryfikować przy pomocy słownika. Katalog powinien zawierać pliki odpowiadające nazwom projektów. Przykładowa sesja:

C:\Users\jan>check Kowalski Jan.zip

Archiwum niepoprawne, nie zawiera katalogu Kowalski Jan.

Projekt 9 (5.0)

Napisz program shell zgodnie z opisem w pliku shell.docx.

Projekt 10 (5.0)

Napisz program sort zgodnie z opisem w pliku sort.docx.

Projekt 11 (5.0)

Napisz program sudoku zgodnie z opisem w pliku sudoku.docx.

Projekt 12 (5.0)

Napisz w języku Java program sudoku zgodnie z opisem w pliku sudoku.docx.

Projekt 13 (5.0)

Napisz program process-tree, który wyświetla strukturę procesów systemu w formie drzewa. Każdy węzeł drzewa powinien zawierać ID procesu, nazwę procesu, oraz ID procesu rodzica.

$ ./process-tree

1 systemd

├─ 450 bash

│ ├─ 1053 vim

│ └─ 1204 top

└─ 452 sshd

└─ 460 bash

Projekt 14 (5.0)

Napisz program mem-manager, który demonstruje dynamiczne zarządzanie pamięcią. Program powinien umożliwiać użytkownikowi alokację, zwalnianie i ponowną alokację pamięci przy pomocy funkcji malloc, realloc, i free. W programie należy zapewnić monitoring użycia pamięci i informować użytkownika o fragmentacji.

$ ./mem-manager

Alokacja: 100 B - Success

Alokacja: 200 B - Success

Zwolnienie: 100 B - Success

Fragmentacja: 33%

Projekt 15 (5.0)

Napisz program env-manager, który wyświetla, dodaje lub usuwa zmienne środowiskowe na poziomie sesji. Program powinien akceptować polecenia list, add <name>=<value>, oraz remove <name>.

$ ./env-manager list

PATH=/usr/bin:/bin

HOME=/home/user

$ ./env-manager add TEST\_VAR=hello

Added TEST\_VAR

$ ./env-manager list

PATH=/usr/bin:/bin

HOME=/home/user

TEST\_VAR=hello

$ ./env-manager remove TEST\_VAR

Removed TEST\_VAR

Projekt 16 (4.5)

Napisz program file-search, który przeszukuje system plików od podanego katalogu w poszukiwaniu plików o określonym rozszerzeniu lub zawierających określony ciąg znaków. Program powinien wyświetlać pełną ścieżkę do każdego znalezionego pliku.

$ ./file-search /home/user .txt

/home/user/documents/report.txt

/home/user/notes/todo.txt

Projekt 17 (3.5)

Napisz program file-permissions, który umożliwia zmianę uprawnień do plików. Program powinien przyjmować nazwę pliku oraz nowy zestaw uprawnień w systemie notacji oktalnej, np., 755, 644. Wyświetla również aktualne uprawnienia.

$ ./file-permissions report.txt 644

Updated permissions to 644

Current permissions: rw-r--r--

Projekt 18 (3.5)

Napisz program time-calc, który wykonuje operacje arytmetyczne na datach. Program przyjmuje datę początkową oraz liczbę dni do dodania lub odjęcia i wyświetla wynikową datę.

$ ./time-calc "2024-01-01" +30

Resulting date: 2024-01-31

$ ./time-calc "2024-01-01" -15

Resulting date: 2023-12-17

Projekt 19 (4.5)

Napisz program user-info, który wyświetla szczegółowe informacje o użytkowniku systemu. Program powinien pobierać informacje takie jak: UID, GID, pełna nazwa, katalog domowy, oraz aktualna powłoka.

$ ./user-info jkowalski

Username: jkowalski

UID: 1001

GID: 1001

Full Name: Jan Kowalski

Home Directory: /home/jkowalski

Shell: /bin/bash

Projekt 20 (4.0)

Napisz program sysinfo, który wyświetla różne informacje systemowe, takie jak użycie pamięci, czas pracy systemu, liczba aktywnych procesów oraz liczba połączeń sieciowych. Program powinien działać na systemie Linux.

$ ./sysinfo

Memory Usage: 2.3 GB / 8.0 GB

Uptime: 1 day, 2 hours, 45 minutes

Active Processes: 123

Network Connections: 15

Projekt 21 (4.5)

Napisz program histogram, który czyta plik tekstowy i wyświetla histogram częstości występowania poszczególnych znaków. Program powinien ignorować wielkość liter oraz pomijać znaki nieliterowe.

$ ./histogram sample.txt

a: \*

b:

c:

d: \*

...

Projekt 22 (5.0)

Napisz program fifo-chat, który implementuje prosty system czatu między dwoma procesami za pomocą kolejek FIFO. Każdy proces powinien mieć swoją kolejkę FIFO do odbierania wiadomości.

# Terminal 1

$ ./fifo-chat send

Connected to chat. Type your message:

> Hello from terminal 1!

# Terminal 2

$ ./fifo-chat receive

Received: Hello from terminal 1!

Projekt 23 (4.5)

Napisz program path-resolver, który zamienia ścieżki względne na bezwzględne oraz rozwija symbole . i ... Program powinien również obsługiwać zmienne środowiskowe w ścieżkach, np. $HOME.

$ ./path-resolver ./../home/$USER/docs

Resolved path: /home/user/docs

Projekt 24 (5.0)

Napisz program virtual-memory-monitor, który pokazuje zużycie pamięci wirtualnej przez każdy proces w systemie. Program powinien wyświetlać PID, nazwę procesu, i ilość zajętej pamięci wirtualnej.

$ ./virtual-memory-monitor

PID Process Name VM Size

123 firefox 800 MB

234 code 300 MB

456 terminal 50 MB

Projekt 25 (4.5)

Napisz program backup-tool, który tworzy kopie zapasowe katalogów. Program powinien przyjmować katalog źródłowy i docelowy oraz wykonywać kopie zapasowe w cyklicznych odstępach czasu (np. co 24 godziny).

$ ./backup-tool /home/user/docs /backup/docs

Backup completed at: 2024-11-01 00:00:00

Projekt 26 (4.0)

Napisz program resource-monitor, który monitoruje użycie zasobów (CPU, pamięć) przez procesy potomne. Program powinien uruchamiać określony proces i monitorować jego zużycie CPU oraz pamięci w odstępach 1 sekundy, aż proces się zakończy.

$ ./resource-monitor "firefox"

Monitoring process: firefox

CPU Usage: 10% | Memory Usage: 500 MB

...

Projekt 27 (4.5)

Napisz program network-port-scanner, który skanuje otwarte porty na wskazanym hoście. Program powinien przyjmować zakres portów do skanowania oraz wyświetlać otwarte porty.

$ ./network-port-scanner localhost 20 80

Open ports on localhost:

22

80