Laboratorium 1

Zad. 1.1

Napisz program InputOutput odczytujący z konsoli i wypisujący na niej kolejno wartości typu int i float. Przykładowa sesja:

a = 5

int value a = 5

b = 5

float value b = 5.000000

Zad. 1.1.1 \*

Do programu dodaj kod, który odczyta z konsoli i wypisze na niej wartość zmienno-przecinkową typu double z dokładnością 15 cyfr po przecinku. Przykładowa sesja:

a = 5

int value a = 5

b = 5

float value b = 5.000000

c = 5

float value c = 5.000000000000000

Zad. 1.2

Napisz program InputOutput2 odczytujący i wypisujący na konsoli napis przy pomocy funkcji scanf. Bufor napisu str ma mieć rozmiar 10. Odczyt napisu nie może powodować przepełnienia bufora. Przykładowa sesja:

a = 0123456789

string value a = 012345678

- dlaczego wypisany napis nie ma litery 9 ?

- czy funkcja scanf zapisuje do bufora białe znaki?

Zad. 1.2.1

Do programu dodaj kod, który odczyta z konsoli napis abcdefghij przy pomocy funkcji fgets z wykorzystaniem poprzedniego bufora napisu str oraz wypisze go na konsoli. Przykładowa sesja:

a = 0123456789

string value a = 012345678

b = string value b = 9

- dlaczego nie można pobrać drugiego napisu?

- dlaczego na konsoli wyświetla się literka 9?

- w jaki sposób wyczyścić zawartość strumienia wejściowego?

ChatGPT: How can I flush stdin in C on Linux?

- popraw program dodając do niego funkcję flush\_stdin()

Zad. 1.3 \*

Napisz program scanf-test sprawdzający, czy funkcja scanf zapisuje do bufora białe znaki i terminator łańcucha. Przetestować spacje, tabulacje i przejście do nowej linii. Przykładowa sesja:

str = ala ma kota

str = ala\0

Po pierwszym napisie został dwa razy wciśnięty klawisz tabulacji oraz Enter.

Zad. 1.4 \*

W programie InputOutput3 odczytaj i wypisz na konsoli cały napis "ala ma kota" przy pomocy funkcji scanf.

Zad. 1.5 \*

W programie InputOutput4 odczytaj i wypisz na konsoli cały napis "ala ma kota" przy pomocy funkcji fgets.

- czy funkcja fgets zapisuje do bufora białe znaki?

Zad. 1.6 \*

Napisz program fgets-test sprawdzający, czy funkcja fgets zapisuje do bufora białe znaki i terminator łańcucha. Przetestować spacje, tabulacje i przejście do nowej linii. Przykładowa sesja:

str = ala ma kota

str = ala ma kota\t\t\n\0

Po pierwszym napisie został dwa razy wciśnięty klawisz tabulacji oraz Enter.

Zad. 1.7 \*

Napisz program stdin-test sprawdzający czy strumień wejściowy stdin jest pusty dla systemu Windows i Linux.

Zad. 1.8 \*

W pliku fibo.txt podaj słowną oraz rekurencyjną definicję ciągu Fibonacciego.

0 1 2 3 4 5 6 indeksy

1 1 2 3 5 8 13 wartosci

Zad. 1.9

Napisz program Fibo wyliczający wartości ciągu Fibonacciego przy pomocy trzech funkcji.

Zad. 1.9.1

Funkcja fibo1 - metoda dziel i zwyciężaj.

f(0) = 1

f(1) = 1

f(n) = f(n-1) + f(n-2)

- dokonaj analizy wywołania fibo1(4).

- narysuj drzewo wywołań dla fibo1(4).

Zad. 1.9.2

Funkcja fibo2 - metoda programowania dynamicznego z ramką trójzębną.

r0 r1 r2

|---|---| .

0 1 2 3 4 5 6 indeksy

1 1 2 3 5 8 13 wartości

|---|---|

r0 r1 r2

Przesunięcie ramki w prawo:

r0 = r1

r1 = r2

r2 = r1 + r0

- ile razy należy przesunąć ramkę w prawo, aby wyznaczyć wartość n-tego wyrazu ciągu Fibonacciego w funkcji fibo2 dla n >= 3 ?

- dokonaj analizy wywołania fibo2(4).

- narysuj graf obliczeń dla fibo2(4).

Zad. 1.9.3 \*

Funkcja fibo3 - metoda programowania dynamicznego z ramką dwuzębną.

r0 r1

|---| .

0 1 2 3 4 5 6 indeksy

1 1 2 3 5 8 13 wartości

| |---|

pom r0 r1

Przesunięcie ramki w prawo:

pom = r0

r0 = r1

r1 = r0 + pom

- ile razy należy przesunąć ramkę w prawo, aby wyznaczyć wartość n-tego wyrazu ciągu Fibonacciego w funkcji fibo3 dla n >= 2 ?

- dokonaj analizy wywołania fibo3(4).

- narysuj graf obliczeń dla fibo3(4).

- która funkcja ma mniejszą złożoność obliczeniową fibo2 czy fibo3 ?

Przykładowa sesja:

fibo1(4) = 5

fibo2(4) = 5

fibo3(4) = 5

Zad. 1.10 \*

Podaj cztery inne funkcje wyliczające rekurencyjnie wartości ciągu Fibonacciego.

Zad. 1.11 \*

Napisz program FiboTree wypisujący, jak wyglądają kolejne wywołania funkcji fibo1 razem z wartościami przez nie zwracanymi. Przykładowa sesja:

fibo1(4) = 5

fibo2(3) = 3

fibo3(2) = 2

fibo4(1) = 1

fibo5(0) = 1

fibo6(1) = 1

fibo7(2) = 2

fibo8(1) = 1

fibo9(0) = 1

- sprawdź czy drzewo wywołań z wcześniejszego zadania zostało poprawnie narysowane

Zad. 1.12 \*

W pliku liczba.txt wyznacz funkcję f(n) wyliczającą liczbę cyfr dla dowolnej liczby całkowitej n. Wykorzystaj funkcję log10(x).

Laboratorium 2

Zad. 2.1 \*

Napisz program Sequence wyliczający wartości ciągu {an} przy pomocy trzech funkcji. Ciąg zdefiniowany jest rekurencyjnie:

a(0) = 1

a(1) = 4

a(n) = 2\*a(n-1) + 0.5\*a(n-2)

- wylicz dziesięć pierwszych wyrazów ciągu {an} w programie Excel

Zad. 2.1.1 \*

Funkcja a1 - metoda dziel i zwyciężaj.

- dokonaj analizy wywołania a1(4).

- narysuj drzewo wywołań dla a1(4).

Zad. 2.1.2 \*

Funkcja a2 - metoda programowania dynamicznego z ramką trójzębną.

- narysuj schemat dla ramki trójzębnej analogicznie jak dla ciągu Fibonacciego

- ile razy należy przesunąć ramkę w prawo, aby wyznaczyć wartość n-tego wyrazu ciągu {an} w funkcji a2 dla n >= 3 ?

- pętla przesuwająca ramkę tym razem musi startować od indeksu 3

- dokonaj analizy wywołania a2(4).

- narysuj graf obliczeń dla a2(4).

Zad. 2.1.3 \*

Funkcja a3 - metoda programowania dynamicznego z ramką dwuzębną.

- narysuj schemat dla ramki dwuzębnej analogicznie jak dla ciągu Fibonacciego

- ile razy należy przesunąć ramkę w prawo, aby wyznaczyć wartość n-tego wyrazu ciągu {an} w funkcji a3 dla n >= 2 ?

- pętla przesuwająca ramkę tym razem musi startować od indeksu 2

- dokonaj analizy wywołania a3(4).

- narysuj graf obliczeń dla a3(4).

- która funkcja ma mniejszą złożoność obliczeniową a2 czy a3 ?

Przykładowa sesja:

a1(4) = 42.250000

a2(4) = 42.250000

a3(4) = 42.250000

Zad. 2.2 \*

Napisz program SequenceTree wypisujący, jak wyglądają kolejne wywołania funkcji a1 razem z wartościami przez nie zwracanymi. Przykładowa sesja:

a1(4) = 42.250000

a2(3) = 19.000000

a3(2) = 8.500000

a4(1) = 4.000000

a5(0) = 1.000000

a6(1) = 4.000000

a7(2) = 8.500000

a8(1) = 4.000000

a9(0) = 1.000000

- sprawdź czy drzewo wywołań z wcześniejszego zadania zostało poprawnie narysowane

Zad. 2.3 \*

Zaproponuj metodę kodowanie drzewa katalogowego przy pomocy tablic trójwymiarowych w języku Java. Przetestuj ją dla następujących drzew.

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

/ / \ \ / | \

B C D f1 Y Z f1

/ \ \ /| |\

D f1 E X f1 Z f1

Laboratorium 3

Zad. 3.1

Napisz program Fractions wedle poniższego schematu i przetestuj jego działanie. Rozbuduj program o pozostałe działania na ułamkach, sugerowane nazwy funkcji: sum, sub, mul, quo. Przed każdą dodaną funkcją należy podać wyprowadzenie dla działania.

#include <stdio.h>

#include <stdlib.h>

/\*

gcc Fractions.c -o Fractions -Wall

\*/

struct Fraction {

int num;

int den;

};

/\*

a/b + c/d = (a\*d)/(b\*d) + (c\*b)/(d\*b) = (a\*d + c\*b) / (b\*d)

\*/

struct Fraction sum(struct Fraction x, struct Fraction y) {

int a = x.num;

int b = x.den;

int c = y.num;

int d = y.den;

struct Fraction z;

z.num = a\*d + c\*b;

z.den = b\*d;

return z;

}

int main(int argc, char \*argv[]) {

struct Fraction x = {2, 3};

struct Fraction y = {1, 4};

struct Fraction z = sum(x, y);

printf("%d/%d + %d/%d = %d/%d\n", x.num, x.den,

y.num, y.den, z.num, z.den);

return 0;

}

Zad. 3.1.1

Dodaj do programu funkcję print wypisującą działanie dla podanych ułamków i operatora.

void print(struct Fraction x, struct Fraction y, const char op);

Przykładowa sesja:

2/3 + 1/4 = 11/12

2/3 - 1/4 = 5/12

2/3 \* 1/4 = 2/12

2/3 / 1/4 = 8/3

2/3 : 1/4 = 8/3

$ - nieznane dzialanie

Zad. 3.1.2 \*

Dodaj do programu funkcję printFraction wypisującą ułamek wedle schematu:

printFraction((struct Fraction){2,0}); // NaN

printFraction((struct Fraction){0,2}); // 0

printFraction((struct Fraction){2,4}); // 1/2

printFraction((struct Fraction){-1,2}); // -1/2

printFraction((struct Fraction){1,-2}); // -1/2

printFraction((struct Fraction){4,-2}); // -2

printFraction((struct Fraction){5,-2}); // -2 1/2

Prototyp funkcji:

void printFraction(struct Fraction x);

Zad. 3.2

Napisz program Strings implementujący i testujący następujące funkcje:

Zad. 3.2.1

Funkcja indexOf działa analogicznie do tej samej funkcji z klasy String w języku Java. W implementacji należy wykorzystać zmienne z poniższego schematu.

/\* index

|

0 1 2 3 4 indexes

str -> ['9']['9']['$'][' ']['\0']

| |

str ptr pointers

\*/

int indexOf(const char \*str, int c);

Zad. 3.2.2

Funkcja isNumber sprawdza, czy napis s przechowuje liczbę całkowitą. Wykorzystaj funkcję isdigit z biblioteki standardowej.

/\*

0 1 2 3 indexes

s -> ['-']['3']['5']['\0']

| | | |

s s+1 s+2 s+3 pointers

\*/

int isNumber(const char \*s);

Zad. 3.2.3

Funkcja trim usuwa z napisu s początkowe i końcowe znaki białe. Wykorzystaj funkcję isspace z biblioteki standardowej.

/\*

i i' j' j

0 1 2 3 4 5 6 7

s -> [' '][' ']['a']['l']['a'][' '][' ']['\0']

k

\*/

char \*trim(char \*s);

Zad. 3.2.4 \*

Funkcja identity zwracająca imię i nazwisko oddzielone spacją.

char \*identity(const char \*name, const char \*surname);

Zad. 3.2.5 \*

Funkcja login tworząca login użytkownika na podstawie pierwszej litery imienia i całego nazwiska. Należy pamiętać, że login na systemie Linux składa się z maksymalnie 32 znaków.

char \*login(const char \*name, const char \*surname);

Zad. 3.3 \*

Podaj pisemny opis funkcji rekurencyjnej, która utworzy drzewo katalogowe dla tablicy trójwymiarowej z zad. 2.3 \* z poprzednich laboratoriów. Można użyć funkcji tworzących plik i katalog oraz funkcji zmiany katalogu i przejścia do katalogu poziom wyżej.

Laboratorium 4

Zad. 4.1

Napisz pełną implementację programu Fractions z odczytem danych z terminala i obsługą błędów.

Zad. 4.1.1

Dodaj funkcję getOperator określającą operator z napisu s. Funkcja zwraca prawdę, jeśli napis jest poprawnym operatorem arytmetycznym.

int getOperator(char \*op, const char \*s);

Zad. 4.1.2

Dodaj funkcję getFraction określającą strukturę ułamkową z napisu s. Funkcja zwraca prawdę, jeśli napis jest ułamkiem zwykłym lub liczbą całkowitą. Napis może mieć postać:

liczba

liczba / liczba

/\* index

|

0 1 2 3 4 indexes

s -> ['3']['7']['/']['5']['\0']

| |

s slash pointers

\*/

int getFraction(struct Fraction \*x, const char \*s);

Zad. 4.1.3

Zaimplementuj odczyt z terminala zgodnie z poniższym przykładem. Jeśli wprowadzony napis ma niepoprawny format, to jego odczyt należy powtórzyć. Przykładowa sesja:

a/b = 2/3

c/d = 1/4

op = +

2/3 + 1/4 = 11/12

Zad. 4.1.4 \*

W programie Fractions2 zaimplementuj bezpieczny odczyt z terminala bez możliwości przepełnienia bufora linii.

Laboratorium 5

Zad. 5.1

Przepisz program Files otwierający plik do odczytu oraz implementujący i testujący następujące funkcje:

Zad. 5.1.1

Funkcja printChars odczytuje zawartość pliku bajt po bajcie.

void printChars(FILE \*fp);

Zad. 5.1.2

Funkcja printLines odczytuje zawartość pliku linia po linii.

void printLines(FILE \*fp);

Zad. 5.1.3 \*

Funkcja copy kopiuje pliki analogicznie do komendy cp.

void copy(const char \*addr1, const char \*addr2);

Zad. 5.2

Napisz program CountWords odczytujący plik tekstowy oraz obliczający liczbę słów w tym pliku przy pomocy funkcji:

Zad. 5.2.1

Funkcja countWords1 oblicza liczbę słów przy pomocy zmiennej stanu inside, gdzie wartość 0 oznacza, że zamierzamy odczytać znak poza słowem, a wartość 1 oznacza, że zamierzamy odczytać znak w słowie.

int countWords1(FILE \*fp);

Zad. 5.2.2 \*

Funkcja countWords2 oblicza liczbę słów bez pomocy zmiennej stanu.

int countWords2(FILE \*fp);

Zad. 5.3

Napisz program DisplayWords odczytujący plik tekstowy oraz wypisujący poszczególne słowa tego pliku przy pomocy funkcji strtok z biblioteki standardowej.

void printWords(FILE \*fp);

Zad. 5.4

Napisz program NewLine testujący jakie znaki określają koniec linii w pliku tekstowym na systemie Windows i Linux. Porównaj działanie programu na obu systemach. Zaimplementuj funkcje:

Zad. 5.4.1

Funkcja printHex wypisuje plik w postaci dwucyfrowych liczb szesnastkowych.

void printHex(FILE \*fp);

Zad. 5.4.2

Funkcja printChar wypisuje plik w postaci znaków umieszczonych w apostrofach.

void printChar(FILE \*fp);

Zad. 5.4.3

Funkcja printLinesHex wypisuje plik linia po linii analogicznie do printHex.

void printLinesHex(FILE \*fp);

Zad. 5.4.4

Funkcja printLinesChar wypisuje plik linia po linii analogicznie do printChar.

void printLinesChar(FILE \*fp);

Przykładowa sesja:

1 linux.txt

2 windows.txt

Choose file: 1

linux.txt

61 6C 61 0A 6B 6F 74 0A

'a' 'l' 'a' '\n' 'k' 'o' 't' '\n'

61 6C 61 0A

6B 6F 74 0A

'a' 'l' 'a' '\n'

'k' 'o' 't' '\n'

Pliki do testów można pobrać ze strony:

http://balois.pl/file/pliki.zip

- dlaczego rozmiary tych plików są różne?

Dla systemu Linux program można uruchomić na stronie:

https://cocalc.com

- jakie znaki określają koniec linii w systemie Linux, Windows i Mac OS?

https://pl.wikipedia.org/wiki/Koniec\_linii

Zad. 5.5 \*

Obsługa plików i katalogów w języku C dla systemu Windows. Referat dla dwóch osób na za dwa tygodnie.

Zad. 5.6 \*

Obsługa plików i katalogów w języku C dla systemu Linux. Referat dla dwóch osób na za trzy tygodnie.

Zad. 5.7 \*

Obsługa plików i katalogów w języku Java. Referat dla dwóch osób na za cztery tygodnie.

Laboratorium 6

Zad. 6.1

1. Sprawdzić czy zainstalowany jest kompilator GCC?

2. Zainstaluj GCC jeśli nie jest zainstalowany.

3. Zaloguj się na drugim terminalu ALT+F2.

4. W katalogu domowym utwórz katalog programy.

5. W katalogu programy wykonaj nano hello.c

6. Napisz program HelloWorld i wyjdź z edytora.

Zad. 6.2

1. Na pierwszym terminalu sprawdź czy w katalogu domowym istnieje plik .nanorc

2. Sprawdź w jaki sposób w edytorze Nano ustawić rozmiar tabulacji na 4 i włączyć opcje tworzenia tabulacji z użyciem spacji.

3. W katalogu domowym otwórz plik .nanorc do edycji w edytorze Nano.

4. Dokonaj konfiguracji rozmiaru tabulacji oraz opcji tabulacji za pomocą spacji.

5. Na drugim terminalu otwórz plik hello.c i sprawdź, czy działają nowe ustawienia.

6. Na pierwszym terminalu przejdź do katalogu programy i skompiluj hello.c

gcc hello.c -o hello

7. Uruchom program hello.

8. Dla pliku hello.c ustaw uprawnia tylko do odczytu. \*

9. Kiedy można uruchomić program hello przez podanie jego nazwy?

Zad. 6.3

Przy pomocy komendy whoami odczytaj nazwę bieżącego użytkownika.

Zad. 6.4

Przy pomocy komendy getent passwd odczytaj wpis dla bieżącego użytkownika.

Zad. 6.5

Przy pomocy komendy getent group odczytaj identyfikator grupy staff.

Zad. 6.6

Przy pomocy komendy delgroup usuń grupę staff.

Zad. 6.7

Przy pomocy komendy addgroup utwórz grupę staff z poprzednim identyfikatorem.

- w pliku /etc/group znajdź wpis dla grupy staff

- jaki identyfikator ma grupa staff ?

https://www.computerhope.com/unix/adduser.htm

Zad. 6.8

Przy pomocy komendy adduser do grupy staff dodaj użytkownika dyrektor.

- w pliku /etc/passwd znajdź wpis dla użytkownika dyrektor

- jaki katalog domowy ma użytkownik dyrektor ?

- jaka powłoka obsługuje terminal użytkownika dyrektor ?

Zad. 6.9

Przy pomocy komendy adduser do grupy student dodaj użytkownika jkowalski z opisem jan kowalski oraz hasłem haslo1.

- w pliku /etc/passwd przejrzyj wpis dla użytkownika jkowalski

- jaki katalog domowy ma użytkownik jkowalski ?

- jaka powłoka obsługuje terminal użytkownika jkowalski ?

Zad. 6.10

Przy pomocy komendy deluser usuń użytkownika jkowalski.

Zad. 6.11

Usuń katalog domowy użytkownika jkowalski.

- jaka opcja dla komendy deluser usuwa użytkownika razem z jego katalogiem domowym?

Zad. 6.12

Przy pomocy komendy openssl passwd –crypt zaszyfruj napis haslo1. Powtórz komendę trzy razy. Czy szyfrowanie jest jednoznaczne?

- ile znaków maksymalnie może mieć hasło dla tej komendy?

Zad. 6.13

Przy pomocy komendy useradd do grupy student dodaj użytkownika jkowalski z opisem jan kowalski oraz hasłem haslo1. Parametry dla komendy należy dobrać w taki sposób, aby użytkownik został utworzony analogicznie jak w zadaniu 6.9.

- w pliku /etc/passwd przejrzyj wpis dla użytkownika jkowalski

- jaki katalog domowy ma użytkownik jkowalski ?

- jaka powłoka obsługuje terminal użytkownika jkowalski ?

https://www.computerhope.com/unix/useradd.htm

Zad. 6.14

Przy pomocy komendy userdel usuń użytkownika jkowalski razem z jego katalogiem domowym.

Zad. 6.15

Przepisz i przeanalizuj program crypt-gnu.c ilustrujący szyfrowanie haseł.

http://balois.pl/file/crypt-gnu.c

Zad. 6.16 \*

Przepisz i przeanalizuj program crypt-xopen.c ilustrujący szyfrowanie haseł.

http://balois.pl/file/crypt-xopen.c

Poczytaj strony:

https://linux.die.net/man/3/crypt\_r

https://manpages.ubuntu.com/manpages/bionic/pl/man3/crypt.3.html

Zad. 6.17

Napisz program addusers tworzący użytkowników na podstawie bazy w pliku baza.txt zawierającej:

jan;kowalski;haslo1

piotr;nowak;haslo2

grzegorz;adamski;haslo3

Użytkowników tworzymy analogicznie jak w zadaniu 6.13. Program ma działać poprawnie dla pliku baza.txt zapisanego zarówno w formacie Linux, jak i Windows.

Zad. 6.18

Wejdź na poniższą stronę i obejrzyj program:

https://dpaste.com/HJKBNEDZT

Pobierz program na Linux komendą:

wget dpaste.com/HJKBNEDZT.txt -O hello.c

- jakie rozszerzenie podajemy po adresie strony z programem?

- jaka opcja służy do określenia nazwy pobieranego programu?

Wyświetl, skompiluj i uruchom pobrany program.

Zad. 6.19

Umieść na stronie dpaste.com program addusers i pobierz go na Linux.

Zad. 6.20 \*

Napisz program delusers usuwający użytkowników podanych na liście w pliku.

Laboratorium 7

Zad. 7.1

Napisz program check sprawdzający na jakim systemie został skompilowany. Wykorzystać kompilację warunkową. Przykładowa sesja:

Program compiled on: Windows

Zad. 7.2

Napisz program arguments wypisujący kolejno:

- wartość zmiennej argc

- adres programu

- nazwę programu \*

- przekazane parametry

Przykładowa sesja:

argc = 2

addr: D:\Dev-C\arguments\arguments.exe

name: arguments.exe

params: \*.c

Zad. 7.3

Napisz program list wywołujący komendę dir dla systemu Windows lub ls dla systemu Linux. Parametry podane przy uruchomieniu programu są przekazywane do obu komend.

Zad. 7.4

Napisz program cyfry wyliczający liczbę cyfr danej liczby całkowitej przy pomocy funkcji:

- funkcja cyfry1 wykorzystuje funkcję log10. \*

- funkcja cyfry2 wykorzystuje funkcję snprintf.

- funkcja cyfry3 wykorzystuje dzielenie przez 10. \*

Przykładowa sesja:

cyfry1(-3579) = 4

cyfry2(-3579) = 4

cyfry3(-3579) = 4

Zad. 7.5 \*

Napisz program version wypisujący wersję biblioteki libc na 3 różne sposoby dla systemu Linux i Windows.

Laboratorium 8

Zad. 8.1 \*

Napisz program regexp ilustrujący wyrażenia regularne POSIX’a.

Zad. 8.2

Napisz program misc implementujący i testujący następujące funkcje:

1. Funkcja losuj zwraca wartość losową z przedziału [a, b)

int losuj(int a, int b);

2. Funkcja wariacje1 wylicza metodą kombinatoryczną liczbę ciągów 2-elementowych, jakie można utworzyć ze znaków określonych wyrażeniem regularnym [a-zA-Z0-9./]

int wariacje1();

3. Funkcja wariacje2 wylicza metodą brutalnej siły liczbę ciągów 2-elementowych, jakie można utworzyć z wartości określonych wyrażeniem regularnym [a-zA-Z0-9./] \*

int wariacje2();

4. Funkcja set w sposób losowy określa wartość parametru salt dla funkcji crypt\_r

void set(char salt[2]);

- jaką ma postać i ile wartości może przyjmować parametr salt dla funkcji crypt\_r ?

https://www.man7.org/linux/man-pages/man3/crypt\_r.3.html

- czy dwuelementowego stringa można użyć do zainicjowania tablicy salt[2] ?

5. Funkcja errnoExample ilustruje wypisanie błędu otwarcia pliku przy pomocy zmiennej globalnej errno

void errnoExample();

- przeczytaj rozdział NOTES ze strony:

https://man7.org/linux/man-pages/man3/errno.3.html

- sprawdź na terminalu jaka nazwa symboliczna i jaki opis odpowiada błędowi nr 2.

6. Funkcja perrorExample ilustruje wypisanie błędu otwarcia pliku przy pomocy funkcji perror \*

void perrorExample();

Projekt 1

Napisz w asemblerze i w języku C program ilustrujący wywołanie funkcji systemowej (ang. system call). W programie w C należy użyć wrapera funkcji systemowej lub funkcji syscall, jeśli funkcja systemowa nie ma wrapera. Jedna połowa grupy wykonuje projekt dla kodu 32 bitowego, a druga połowa grupy wykonuje projekt dla kodu 64 bitowego. Sugerowana nazwa programu to nazwa funkcji systemowej.

Deadline: 10.12.2024

Projekty należy wysłać na maila w spakowanym katalogu:

Nazwisko Imie

nazwa.c

nazwa.asm

nazwa.docx ; dokumentacja i raport z testów

W nazwach katalogu i plików nie używamy polskich znaków. Katalog należy spakować programem 7-Zip i sprawdzić przed wysłaniem programem check.

Projekt 2

Projekty należy wykonać w języku C. W programie nie można korzystać z komend powłoki, jeśli analogiczna funkcjonalność może być zrealizowana przy pomocy funkcji z biblioteki libc lub POSIX. 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: 19.01.2025

Projekty należy wysłać z maila uczelnianego w spakowanym katalogu:

Nazwisko Imie

nazwa.c

nazwa.docx ; dokumentacja i raport z testów

W nazwach katalogu i plików nie używamy polskich znaków. Katalog należy spakować programem 7-Zip i sprawdzić przed wysłaniem programem check.

Projekt 1 (3.0)

Napisz program file-permissions, który umożliwia zmianę uprawnień do plików. Program przyjmuje nazwę pliku i nowy zestaw uprawnień w notacji oktalnej oraz wyświetla poprzednie i uaktualnione uprawnienia w notacji symbolicznej. Przykładowa sesja:

$ ./file-permissions report.txt 644

Previous permissions: rwxrwxrwx

Current permissions: rw-r--r--

Projekt 2 (3.0)

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 3 (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. Przykładowe sesje:

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

Resulting date: 2024-01-31

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

Resulting date: 2023-12-17

Projekt 4 (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 5 (3.5)

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 6 (3.5)

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: 8

b: 7

d: 5

...

Projekt 7 (3.5)

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

$ ./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 8 (4.0)

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

$ ./resource-monitor "firefox"

Monitoring process: firefox

CPU Usage: 10% | Memory Usage: 500 MB

...

Projekt 9 (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 10 (4.0)

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. Przykładowa sesja:

# 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 11 (4.5)

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

./file-search /home/user -e txt -s Hello  
Znaleziono plik: /home/ur/dokumenty/plik.txt

./file-search /var/log -e log  
Znaleziono plik: /var/log/system.log  
Znaleziono plik: /var/log/debug.log

./file-search . -s Error  
Znaleziono plik: /home/user/folder/bledy.txt

Projekt 12 (4.5)

Napisz program path-resolver, który zamienia adresy względne na bezwzględne. Program powinien brać pod uwagę symbole . i .. oraz zmienne środowiskowe, np. $HOME. Przykładowa sesja:

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

Resolved path: /home/user/docs

Projekt 13 (4.5)

Napisz program backup-tool, który tworzy kopie zapasowe katalogów. Program powinien przyjmować katalog źródłowy i docelowy oraz wykonywać kopie zapasowe cyklicznie, np. co 24 godziny. Przykładowa sesja:

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

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

Projekt 14 (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

Projekt 15 (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 16 (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 17 (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 18 (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 19 (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 20 (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 21 (5.0)

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

Projekt 22 (5.0)

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

Projekt 23 (5.0)

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

Projekt 24 (5.0)

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

Projekt 25 (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. Przykładowa sesja:

1 systemd

├─ 450 bash

│ ├─ 1053 vim

│ └─ 1204 top

└─ 452 sshd

└─ 460 bash

Dla powyższego przykładu wyświetlamy:

1 systemd

+- 450 bash

| +- 1053 vim

| +- 1204 top

+- 452 sshd

+- 460 bash

Projekt 26 (5.0)

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

PID Process Name VM Size

123 firefox 800 MB

234 code 300 MB

456 terminal 50 MB

Laboratorium 9

Zad. 9.1

Przepisz program copy oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define MAX 512

int main(int argc, char\* argv[]) {

char buf[MAX];

int desc\_zrod, desc\_cel;

int lbajt;

if (argc < 3) {

fprintf(stderr, "Za malo argumentow. Uzyj:\n");

fprintf(stderr, "%s <plik zrodlowy> <plik docelowy>\n", argv[0]);

exit(1);

}

desc\_zrod = open(argv[1], O\_RDONLY);

if (desc\_zrod == -1) {

perror("Blad otwarcia pliku zrodlowego");

exit(1);

}

desc\_cel = creat(argv[2], 0640);

if (desc\_cel == -1) {

perror("Blad utworzenia pliku docelowego");

exit(1);

}

while ((lbajt = read(desc\_zrod, buf, MAX)) > 0) {

if (write(desc\_cel, buf, lbajt) == -1) {

perror("Blad zapisu pliku docelowego");

exit(1);

}

}

if (lbajt == -1) {

perror("Blad odczytu pliku zrodlowego");

exit(1);

}

if (close(desc\_zrod) == -1 || close(desc\_cel) == -1) {

perror("Blad zamkniecia pliku");

exit(1);

}

exit(0);

}

- co przechowują parametry: argv[0], argv[1], argv[2] ?

- omów funkcje: open, creat, read, write, close, exit.

Zad. 9.2

Przepisz program size oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

int main(int argc, char\* argv[]) {

int desc;

long rozm;

if (argc < 2) {

fprintf(stderr, "Za malo argumentow. Uzyj:\n");

fprintf(stderr, "%s <nazwa pliku>\n", argv[0]);

exit(1);

}

desc = open(argv[1], O\_RDONLY);

if (desc == -1) {

perror("Blad otwarcia pliku");

exit(1);

}

rozm = lseek(desc, 0, SEEK\_END);

if (rozm == -1) {

perror("Blad w pozycjonowaniu");

exit(1);

}

printf("Rozmiar pliku %s: %ld\n", argv[1], rozm);

if (close(desc) == -1) {

perror("Blad zamkniecia pliku");

exit(1);

}

exit(0);

}

- omów funkcję lseek

Zad. 9.3 \*

Napisz program kopiujący zawartość pliku o nazwie podanej jako pierwszy parametr, do pliku którego nazwa podana jest jako drugi parametr.

Zad. 9.4 \*

Napisz program zmieniający kolejność znaków w każdej linii pliku o nazwie podanej jako parametr.

Zad. 9.5 \*

Napisz procedurę kopiowania ostatnich 10 znaków, słów i ostatnich 10 linii jednego pliku do innego.

Zad. 9.6 \*

Napisz program do rozpoznawania czy plik o podanej nazwie jest plikiem tekstowym, plik tekstowy zawiera znaki o kodach 0-127, można w tym celu użyć funkcji isascii.

Zad. 9.7 \*

Napisz program, który w pliku o nazwie podanej jako ostatni argument zapisze połączoną zawartość wszystkich plików, których nazwy zostały podane w linii poleceń przed ostatnim argumentem.

Zad. 9.8 \*

Napisz program do porównywania plików o nazwach przekazanych jako argumenty. Wynikiem działania programu ma być komunikat, że pliki są identyczne, pliki różnią się od znaku nr <nr znaku> w linii <nr znaku linii> lub - gdy jeden z plików zawiera treść drugiego uzupełnioną o jakieś dodatkowe znaki - plik <nazwa> zawiera <liczba> znaków więcej niż zawartość pliku <nazwa>.

Laboratorium 10

Zad. 10.1

Przepisz program fork oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <unistd.h>

int main() {

printf("Poczatek\n");

fork();

printf("Koniec\n");

return 0;

}

- w jaki sposób działa funkcja fork ?

- dlaczego napis Poczatek wyświetla się tylko raz?

- dlaczego napis Koniec wyświetla się dwa razy?

Zad. 10.2

Przepisz program fork2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

printf("Poczatek\n");

pid\_t pid = fork();

if (pid < 0) fprintf(stderr, "Blad tworzenia procesu potomnego.\n");

if (pid == 0) printf("Proces dziecko, zmienna pid = %d\n", pid);

if (pid > 0) printf("Proces rodzic, zmienna pid = %d\n", pid);

printf("Koniec\n");

return 0;

}

- jaką wartość przechowuje zmienna pid dla procesu potomnego?

- jaką wartość przechowuje zmienna pid dla procesu macierzystego?

- jaki pid ma proces potomny?

- jaki pid ma proces macierzysty?

Zad. 10.3

Napisz program fork3 na podstawie fork2 w taki sposób, aby proces rodzic oczekiwał na zakończenie procesu dziecko.

- który proces wyświetla przedostatni napis Koniec ?

Zad. 10.4

Napisz program fork4 na podstawie fork3 w taki sposób, aby proces rodzic i proces dziecko dodatkowo wyświetlał swój pid przy pomocy funkcji getpid().

Zad. 10.5 \*

Napisz program fork5 w którym proces dziecko wyświetla własny pid oraz pid rodzica. Przykładowa sesja:

Poczatek

Koniec

Proces dziecko, pid dziecka = 3257

Proces dziecko, pid rodzica = 3256

Koniec

- który proces wyświetla napis Koniec jako pierwszy?

Zad. 10.6

Przepisz program execlp oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <unistd.h>

int main() {

printf("Poczatek\n");

execlp("ls", "ls", "-l", NULL);

printf("Koniec\n");

return 0;

}

- omów funkcję execlp

- dlaczego napis Koniec się nie wyświetla?

Zad. 10.7

Przepisz program execlp2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

printf("Poczatek\n");

if (fork() == 0) {

execlp("ls", "ls", "-a", NULL);

perror("Blad uruchmienia programu");

exit(1);

}

wait(NULL);

printf("Koniec\n");

return 0;

}

- dlaczego tym razem napis Koniec się wyświetla?

Zad. 10.8 \*

Przepisz program execlp3 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

printf("Poczatek\n");

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego");

return 1;

}

if (pid == 0) {

execlp("ls", "ls", "-l", NULL);

perror("Blad uruchmienia programu");

exit(1);

}

if (wait(NULL) == -1)

perror("Blad w oczekiwaniu na zakonczenie potomka");

printf("Koniec\n");

return 0;

}

- opisz na podstawie dokumentacji man wartości zwracane przez funkcje fork() i wait()

Zad. 10.9

Przepisz program sierota oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

if (fork() == 0) {

printf("child pid = %d\n", getpid());

sleep(20);

exit(0);

}

exit(0);

}

- w jakim przypadku proces staje się sierotą?

- sprawdź stan programu sierota na liście procesów

- do jakiego procesu został przypisany proces dziecko?

Zad. 10.10

Przepisz program zombie oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

if (fork() == 0) {

printf("child pid = %d\n", getpid());

exit(0);

}

sleep(20);

wait(NULL);

exit(0);

}

- w jakim przypadku proces staje się zombie?

- sprawdź stan programu zombie na liście procesów

Zad. 10.11

Przepisz program status oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

pid\_t pid1 = fork();

if (pid1 == 0) exit(7); // proces dziecko

printf("pid rodzica = %d\n", getpid());

printf("pid dziecka = %d\n\n", pid1);

int status;

pid\_t pid2 = wait(&status);

printf("Proces: %d\n", pid2);

printf("Status: %x\n", status);

}

- w jaki sposób dziecko zwraca status wyjścia do rodzica?

- w jaki sposób rodzic odbiera status wyjścia od dziecka?

- jakiego procesu pid zwraca funkcja wait?

Zad. 10.12

Przepisz program status2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

pid\_t pid1 = fork();

if (pid1 == 0) { // proces dziecko

sleep(10);

exit(7);

}

printf("pid rodzica = %d\n", getpid());

printf("pid dziecka = %d\n\n", pid1);

kill(pid1, 9);

int status;

pid\_t pid2 = wait(&status);

printf("Proces: %d\n", pid2);

printf("Status: %x\n", status);

}

- który proces odbiera sygnał nr 9?

- jaki sygnał reprezentuje liczba 9?

kill -l 9

- wyświetl listę wszystkich sygnałów

kill -l

Zad. 10.13

Przepisz program redirect oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

int main(int argc, char\* argv[]) {

int fd = creat("ls.txt", 0600);

printf("fd = %d\n", fd); // fd = 3

// 1 -> stdout

// 3 -> ls.txt

dup2(fd, 1);

// 1 -> ls.txt

// 3 -> ls.txt

execvp("ls", argv); // ls uses file descriptor 1 to output the result

}

- czy funkcje creat i open alokują deskryptory?

- na podstawie dokumentacji man opisz działanie funkcji creat i open \*

- jakie deskryptory i strumienie są przypisywane do nowego procesu?

ls /dev -l | grep fd

- jaki deskryptor jest zazwyczaj przypisany do standardowego wyjścia stdout ?

- jaki deskryptor został przypisany do pliku ls.txt ?

- ile deskryptorów jest przypisanych do pliku ls.txt po wykonaniu dup2 ?

- który deskryptor w wywołaniu dup2 jest stary a który nowy?

- na czym polega duplikowanie deskryptora przy pomocy funkcji dup2 ?

- w jaki sposób działa funkcja execvp ?

- w jaki sposób wynik programu ls przekierowywany jest do pliku ls.txt ?

https://en.wikipedia.org/wiki/File\_descriptor

https://www.computerhope.com/jargon/f/file-descriptor.htm

Zad. 10.14

Przepisz program redirect2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

int main(int argc, char\* argv[]) {

int fd = creat("ls.txt", 0600);

printf("fd = %d\n", fd); // fd = 3

// 0 -> stdin

// 1 -> stdout

// 2 -> stderr

// 3 -> ls.txt

close(1);

// 0 -> stdin

// 1

// 2 -> stderr

// 3 -> ls.txt

dup(fd); // duplicates fd (3) to the lowest available file descriptor (1)

// 0 -> stdin

// 1 -> ls.txt

// 2 -> stderr

// 3 -> ls.txt

execvp("ls", argv); // ls uses file descriptor 1 to output the result

}

- w jaki sposób działa instrukcja close(1) ?

- w jaki sposób działa instrukcja dup(fd) ?

- jaki plik przypisany jest do deskryptora 1 po wykonaniu funkcji dup ?

- ile i jakie deskryptory są przypisane do pliku ls.txt po wykonaniu dup ?

- w jaki sposób wynik programu ls przekierowywany jest do pliku ls.txt ?

Zad. 10.15

Przepisz program redirect3 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

/\*

gcc redirect3.c -o redirect3 -Wall

\*/

int main(int argc, char\* argv[]) {

// 0 -> stdin

// 1 -> stdout

// 2 -> stderr

close(1);

// 0 -> stdin

// 1

// 2 -> stderr

creat("ls.txt", 0600); // creat assigns ls.txt to the lowest

// available file descriptor, which is 1

// 0 -> stdin

// 1 -> ls.txt

// 2 -> stderr

execvp("ls", argv); // ls uses file descriptor 1 to output the result

}

- w jaki sposób działa instrukcja close(1) ?

- czy funkcja creat alokuje deskryptory?

- jaki deskryptor zostanie przypisany do pliku ls.txt po wykonaniu creat ?

- jaki deskryptor jest zazwyczaj przypisany do standardowego wyjścia stdout ?

- w jaki sposób wynik programu ls przekierowywany jest do pliku ls.txt ?

Zad. 10.16 \*

Napisz program tworzący dwa procesy. Każdy ze stworzonych procesów powinien utworzyć proces - potomka. Należy wyświetlać identyfikatory procesów rodziców i potomków po każdym wywołaniu funkcji fork.

Zad. 10.17 \*

Napisz program tworzący równocześnie trzy procesy zombie.

Zad. 10.18 \*

Napisz program, którego rezultatem będzie wydruk zawartości bieżącego katalogu poprzedzony napisem Poczatek a zakończony napisem Koniec.

Laboratorium 11

Zad. 11.1

Napisz program psize wypisujący rozmiar potoku.

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

int main() {

int pdesk[2];

if (pipe(pdesk) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

printf("pdesk[0] = %d\n", pdesk[0]);

printf("pdesk[1] = %d\n\n", pdesk[1]);

int psize = fcntl(pdesk[0], F\_GETPIPE\_SZ);

printf("psize = %d\n", psize);

return 0;

}

- w jaki sposób działa funkcja pipe ?

- gdzie są deskryptory przypisane do potoku?

- jakie deskryptory zostały przypisane do potoku?

- jaki deskryptor ma koniec potoku do odczytu?

- jaki deskryptor ma koniec potoku do zapisu?

- w jaki sposób można określić rozmiar potoku? \*

Zad. 11.2 \*

Napisz program psize2 wypisujący maksymalną liczbę bajtów, które można zapisać atomowo do potoku.

Zad. 11.3

Omów poniższy schemat potoku nienazwanego.

read <- [3][5][ ][ ] <- write

| |

pd[0] pd[1]

- jaki rozmiar ma ten potok?

- ile bajtów znajduje się w potoku?

- ile otwartych deskryptorów ma ten potok?

- jaki deskryptor ma koniec potoku do odczytu?

- jaki deskryptor ma koniec potoku do zapisu?

- jakich funkcji używamy do odczytu i zapisu?

- ile mamy procesów odczytujących i zapisujących?

Zad. 11.4

Załóżmy, że mamy procesy odczytujące i zapisujące do potoku.

- jakie procesy są blokowane na pustym potoku?

read <- [ ][ ][ ][ ] <- write

| |

pd[0] pd[1]

- jakie procesy są blokowane na pełnym potoku?

read <- [3][5][7][9] <- write

| |

pd[0] pd[1]

Zad. 11.5

Napisz program closepd0 ilustrujący zapis do potoku bez procesów odczytujących.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

void sigHandler(int signum) {

psignal(signum, "Signal SIGPIPE ");

}

int main() {

signal(SIGPIPE, sigHandler); // register signal handler

int pdesc[2];

if (pipe(pdesc) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

close(pdesc[0]);

if (write(pdesc[1], "0", 1) == -1) {

perror("Blad zapisu do potoku ");

exit(1);

}

exit(0);

}

- jaki sygnał jest wzbudzany przy zapisie do potoku bez procesów odczytujących?

- czy zapis do potoku bez procesów odczytujących powoduje błąd?

- co funkcja write próbuje zapisać do potoku?

- czy w funkcji write napis "0" można zastąpić znakiem '0' ?

- jaka sytuację przedstawia poniższy schemat potoku?

[ ][ ][ ][ ] <- write

|

pdesc[1]

- czy potok musi być pusty, aby doszło do scenariusza z powyższego programu?

Zad. 11.6

Napisz program closepd1 ilustrujący odczyt z pustego potoku bez procesów zapisujących.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

int pdesc[2];

if (pipe(pdesc) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

pid\_t pid = fork();

if (pid == 0) {

printf("Parent is blocked for two seconds.\n");

sleep(2);

close(pdesc[1]);

printf("Parent will read 0 bytes and exit.\n");

}

if (pid > 0) {

close(pdesc[1]);

char buf[1];

int bytes = read(pdesc[0], buf, 1);

printf("Parent has read %d bytes and exits.\n", bytes);

}

exit(0);

}

- dlaczego proces rodzic jest blokowany?

- na której instrukcji blokowany jest proces rodzic?

- dlaczego proces rodzic jest blokowany tylko na 2 sekundy?

- czy odczyt z potoku bez procesów zapisujących powoduje błąd?

- jaka sytuację przedstawia poniższy schemat potoku?

read <- [ ][ ][ ][ ]

|

pdesc[0]

- czy potok musi być pusty, aby doszło do scenariusza z powyższego programu?

- w jakim przypadku funkcja read zwraca 0 dla potoku nienazwanego?

Zad. 11.7

Przepisz program pipe oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main () {

int pdesk[2];

if (pipe(pdesk) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

if (write(pdesk[1], "Hallo!", 7) == -1) {

perror("Blad zapisu do potoku ");

exit(1);

}

exit(0);

}

if (pid > 0) {

char buf[10];

if (read(pdesk[0], buf, 10) == -1) {

perror("Blad odczytu z potoku ");

exit(1);

}

printf("Odczytano z potoku: %s\n", buf);

}

return 0;

}

- ile bajtów odczyta z potoku proces rodzic?

Zad. 11.8

Przepisz program pipe2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main() {

int pdesk[2];

pipe(pdesk);

pid\_t pid = fork();

if (pid == 0)

write(pdesk[1], "Hallo!", 7);

if (pid > 0) {

char buf[10];

read(pdesk[0], buf, 10);

read(pdesk[0], buf, 10);

printf("Odczytano z potoku: %s\n", buf);

}

return 0;

}

- na której instrukcji blokuje się program i dlaczego?

- jaką instrukcję należy dodać i gdzie, aby program się nie zablokował?

Zad. 11.9

Przepisz program pipe3 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <unistd.h>

#define MAX 512

int main (int argc, char\* argv[]) {

int pdesk[2];

if (pipe(pdesk) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

dup2(pdesk[1], 1);

execvp("ls", argv);

perror("Blad uruchomienie programu ls ");

exit(1);

}

if (pid > 0) {

close(pdesk[1]);

char buf[MAX];

int lb, i;

while ((lb = read(pdesk[0], buf, MAX)) > 0) {

for (i = 0; i < lb; i++)

buf[i] = toupper(buf[i]);

if (write(1, buf, lb) == -1) {

perror("Blad zapis na standardowe wyjscie ");

exit(1);

}

}

if (lb == -1) {

perror("Blad odczytu z potoku ");

exit(1);

}

}

return 0;

}

- na czym polega duplikowanie deskryptora?

- jaki deskryptor jest zazwyczaj przypisany do standardowego wyjścia stdout ?

- w jaki sposób zostało przekierowane standardowe wyjście procesu ls ?

- do jakiego strumienia zapisuje w programie funkcja write ?

Zad. 11.10

Przepisz program pipe4 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <unistd.h>

#define MAX 512

int main (int argc, char \*argv[]) {

int pdesk[2];

if (pipe(pdesk) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

dup2(pdesk[1], 1);

execvp("ls", argv);

perror("Blad uruchomiania programu ls ");

exit(1);

}

if (pid > 0) {

close(pdesk[1]);

dup2(pdesk[0], 0);

execlp("tr", "tr", "a-z", "A-Z", NULL);

perror("Blad uruchomiania programu tr ");

}

return 0;

}

- jaki deskryptor jest zazwyczaj przypisany do standardowego wejścia stdin ?

- w jaki sposób zostało przekierowane standardowe wejście procesu tr ?

- skąd odczytuje dane proces tr ?

- gdzie zapisuje dane proces tr ?

Zad. 11.11

Przepisz program fifo oraz przeanalizuj jego działanie.

#include <fcntl.h>

#include <sys/stat.h>

int main() {

mkfifo("kolFIFO", 0600);

open("kolFIFO", O\_RDONLY);

return 0;

}

- dlaczego i na jakiej funkcji program się blokuje?

Zad. 11.12

1. Napisz program fwriter ilustrujący zapis do kolejki FIFO.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/stat.h>

int main() {

if (mkfifo("kolFIFO", 0600) == -1) {

perror("Blad tworzenia kolejki FIFO ");

exit(1);

}

int fdesc = open("kolFIFO", O\_WRONLY);

if (fdesc == -1){

perror("Blad otwarcia kolejki do zapisu ");

exit(1);

}

printf("fwriter : fdesc = %d\n", fdesc);

char msg[] = "Message from fwriter.";

write(fdesc, msg, strlen(msg) + 1);

printf("Message sent: %s\n", msg);

return 0;

}

- dlaczego i na jakiej funkcji program się blokuje?

2. Napisz program freader ilustrujący odczyt z kolejki FIFO.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#define MAX 80

int main() {

int fdesc = open("kolFIFO", O\_RDONLY);

if (fdesc == -1) {

perror("Blad otwarcia kolejki do odczytu ");

exit(1);

}

printf("freader : fdesc = %d\n", fdesc);

char buf[MAX];

read(fdesc, buf, MAX);

printf("Message received: %s\n", buf);

return 0;

}

- przetestuj kolejkę FIFO uruchamiając oba programy

- w jakim trybie należy uruchomić program fwriter?

- czym się różni kolejka FIFO od potoku nienazwanego?

Zad. 11.13

Przepisz program fifo2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <unistd.h>

int main() {

int fdesk;

if (mkfifo("/tmp/fifo", 0600) == -1) {

perror("Blad tworzenia kolejki FIFO ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

fdesk = open("/tmp/fifo", O\_WRONLY);

if (fdesk == -1){

perror("Blad otwarcia kolejki do zapisu ");

exit(1);

}

if (write(fdesk, "Hallo!", 7) == -1){

perror("Blad zapisu do kolejki ");

exit(1);

}

exit(0);

}

if (pid > 0) {

fdesk = open("/tmp/fifo", O\_RDONLY);

if (fdesk == -1){

perror("Blad otwarcia kolejki do odczytu ");

exit(1);

}

char buf[10];

if (read(fdesk, buf, 10) == -1){

perror("Blad odczytu z kolejki ");

exit(1);

}

printf("Odczytano z kolejki: %s\n", buf);

}

return 0;

}

Zad. 11.14

Przepisz program fifo3 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <unistd.h>

int main(int argc, char\* argv[]) {

int fdesk;

if (mkfifo("/tmp/fifo", 0600) == -1) {

perror("Blad tworzenia kolejki FIFO ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

close(1);

fdesk = open("/tmp/fifo", O\_WRONLY);

if (fdesk == -1) {

perror("Blad otwarcia kolejki do zapisu ");

exit(1);

}

if (fdesk != 1) {

perror("Niewlasciwy deskryptor do zapisu ");

exit(1);

}

execvp("ls", argv);

perror("Blad uruchmienia programu ");

exit(1);

}

if (pid > 0) {

close(0);

fdesk = open("/tmp/fifo", O\_RDONLY);

if (fdesk == -1) {

perror("Blad otwarcia kolejki do odczytu ");

exit(1);

}

if (fdesk != 0) {

perror("Niewlasciwy deskryptor do odczytu ");

exit(1);

}

execlp("tr", "tr", "a-z", "A-Z", NULL);

perror("Blad uruchomienia programu tr ");

}

return 0;

}

- jaki deskryptor jest zazwyczaj przypisany do standardowego wyjścia stdout ?

- jaki deskryptor zostaje zamknięty w procesie zapisującym?

- w jaki sposób zostało przekierowane standardowe wyjście procesu ls ?

- jaki deskryptor jest zazwyczaj przypisany do standardowego wejścia stdin ?

- jaki deskryptor zostaje zamknięty w procesie odczytującym?

- w jaki sposób zostało przekierowane standardowe wejście procesu tr ?

- skąd odczytuje dane proces tr ?

- gdzie zapisuje dane proces tr ?

Zad. 11.15

Przepisz program pipe3err oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <sys/wait.h>

#include <unistd.h>

#define MAX 512

int main(int argc, char\* argv[]) {

int pdesk[2];

if (pipe(pdesk) == -1) {

perror("Blad tworzenia potoku ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

dup2(pdesk[1], 1);

execvp("ls", argv);

perror("Blad uruchomienie programu ls ");

exit(1);

}

if (pid > 0) {

close(pdesk[1]);

wait(NULL);

char buf[MAX];

int lb, i;

while ((lb = read(pdesk[0], buf, MAX)) > 0) {

for (i = 0; i < lb; i++)

buf[i] = toupper(buf[i]);

if (write(1, buf, lb) == -1) {

perror("Blad zapisu na standardowe wyjscie ");

exit(1);

}

}

if (lb == -1) {

perror("Blad odczytu z potoku ");

exit(1);

}

}

return 0;

}

- w jakim przypadku może dojść do zakleszczenia programu?

Zad. 11.16

Przepisz program fifo3err oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <unistd.h>

#define MAX 512

int main(int argc, char\* argv[]) {

int fdesk;

if (mkfifo("/tmp/fifo", 0600) == -1) {

perror("Blad tworzenia kolejki FIFO ");

exit(1);

}

pid\_t pid = fork();

if (pid == -1) {

perror("Blad tworzenia procesu potomnego ");

exit(1);

}

if (pid == 0) {

close(1);

fdesk = open("/tmp/fifo", O\_WRONLY);

if (fdesk == -1){

perror("Blad otwarcia kolejki do zapisu ");

exit(1);

}

if (fdesk != 1){

perror("Niewlasciwy deskryptor do zapisu ");

exit(1);

}

execvp("ls", argv);

perror("Blad uruchmienia programu ");

exit(1);

}

if (pid > 0) {

close(0);

wait(NULL);

fdesk = open("/tmp/fifo", O\_RDONLY);

char buf[MAX];

int lb, i;

while ((lb = read(fdesk, buf, MAX)) > 0) {

for (i = 0; i < lb; i++)

buf[i] = toupper(buf[i]);

if (write(1, buf, lb) == -1) {

perror("Blad zapisu na standardowe wyjscie ");

exit(1);

}

}

if (lb == -1) {

perror("Blad odczytu z kolejki ");

exit(1);

}

}

return 0;

}

- dlaczego program się zakleszcza?

Zad. 11.17 \*

Napisz program który tworzy trzy procesy - proces macierzysty i jego dwa procesy potomne. Pierwszy z procesów potomnych powinien zapisać do potoku napis „hello!", a drugi proces potomny powinien ten napis odczytać.

Zad. 11.18 \*

Napisz program, który tworzy trzy procesy, z których dwa zapisują do potoku, a trzeci odczytuje z niego i drukuje otrzymane komunikaty.

Zad. 11.19 \*

Napisz programy realizujące następujące potoki:

ls | wc

finger | cut -d' ' -f1

ls -l | grep ^d | more

ps -ef| tr -s ' '  :| cut -d: -f1 |sort| uniq -c |sort n

cat /etc/group | head -5 > grupy.txt

Zad. 11.20 \*

Napisz program tworzący dwa procesy: klienta i serwera. Serwer tworzy ogólnodostępną kolejkę FIFO, i czeka na zgłoszenia klientów. Każdy klient tworzy własną kolejkę, poprzez którą będą przychodzić odpowiedzi serwera. Zadaniem klienta jest przesłanie nazwy stworzonej przez siebie kolejki, a serwera odesłaniem poprzez kolejkę stworzoną przez klienta wyniku polecenia ls.

Zad. 11.21 \*

Zmodyfikować poprzedni program, tak, by kolejka utworzona przez klienta była dwukierunkowa, klient publiczną kolejką powinien przesyłać nazwę stworzonej przez siebie kolejki. Dalsza wymiana komunikatów powinna odbywać się poprzez kolejkę stworzoną przez klienta. Klient kolejką tą powinien wysyłać polecenia, zadaniem serwera jest wykonywanie tych poleceń i odsyłanie wyników.

Laboratorium 12

Zad. 12.1

Przepisz program writer tworzący kolejkę komunikatów, przeanalizuj jego działanie oraz prześlij do kolejki dwa dowolne komunikaty.

#include <stdio.h>

#include <stdlib.h>

#include <sys/msg.h>

#define N 80

struct msgbuf {

long mtype;

char mvalue[N];

};

int main() {

int msgid = msgget(45281, IPC\_CREAT | 0600);

struct msgbuf msg;

printf("Message to be sent: ");

fgets(msg.mvalue, N, stdin);

msg.mtype = 1;

msgsnd(msgid, &msg, sizeof(msg.mvalue), 0);

system("ipcs -q");

return 0;

}

- w jaki sposób tworzymy kolejkę komunikatów?

- jaki typ ma wysyłany komunikat?

- jaki maksymalny rozmiar może mieć wysyłany komunikat?

- jaka komenda podaje informacje o kolejkach komunikatów?

- jaki klucz ma utworzona kolejka komunikatów?

- jaki identyfikator ma utworzona kolejka komunikatów?

- ile komunikatów znajduje się w kolejce?

Zad. 12.2

Przepisz program reader oraz przeanalizuj jego działanie. Odbierz komunikaty przesłane do kolejki i za każdym razem sprawdzaj status kolejki.

#include <stdio.h>

#include <stdlib.h>

#include <sys/msg.h>

#define N 80

struct msgbuf {

long mtype;

char mvalue[N];

};

int main() {

int msgid = msgget(45281, IPC\_CREAT | 0600);

struct msgbuf msg;

msgrcv(msgid, &msg, sizeof(msg.mvalue), 1, 0);

printf("Message received: %s\n", msg.mvalue);

system("ipcs -q");

//msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

- jaki typ ma odbierany komunikat?

- ile komunikatów znajduje się w kolejce?

Zad. 12.3

Przepisz program msq oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/msg.h>

#define N 80

struct msgbuf {

long mtype;

char mvalue[N];

};

int main() {

int msgid = msgget(IPC\_PRIVATE, IPC\_CREAT | 0600);

if (msgid == -1) {

perror("Tworzenie kolejki komunikatow ");

exit(1);

}

pid\_t pid = fork();

if (pid == 0) {

struct msgbuf msg;

msg.mtype = 1;

strcpy(msg.mvalue, "Message from the child.");

printf("Child will send a message in 3 seconds.\n");

sleep(3);

if (msgsnd(msgid, &msg, sizeof(msg.mvalue), 0) == -1) {

perror("Wysylanie komunikatu ");

exit(1);

}

}

if (pid > 0) {

struct msgbuf msg;

if (msgrcv(msgid, &msg, sizeof(msg.mvalue), 1, 0) == -1) {

perror("Odbieranie komunikatu ");

exit(1);

}

printf("Parent received: %s\n", msg.mvalue);

if (msgctl(msgid, IPC\_RMID, NULL) == -1) {

perror("Usuwanie kolejki ");

exit(1);

}

}

exit(0);

}

- co oznacza wartość IPC\_PRIVATE ?

- czy procesy rodzic i dziecko wymagają synchronizacji?

- w jaki sposób usuwamy kolejkę komunikatów?

Zad. 12.4

Przepisz program mwrite tworzący segment pamięci współdzielonej oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

int main() {

int shmid = shmget(45281, MAX\*sizeof(int), IPC\_CREAT | 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczenie segmentu pamieci wspoldzielonej ");

exit(1);

}

printf("mwrite : mem = %p\n\n", mem);

int i;

for (i = 0; i < N; i++) mem[i % MAX] = i;

printf("Data written to shared memory.\n");

shmdt(mem);

system("ipcs -m");

return 0;

}

- w jaki sposób tworzymy segment pamięci współdzielonej?

- w jaki sposób dowiązujemy segment pamięci współdzielonej do procesu?

- jaki jest adres dowiązania segmentu pamięci współdzielonej dla procesu mwrite ?

- w jaki sposób rozpoznajemy błąd dowiązania segmentu pamięci współdzielonej?

- w jaki sposób odłączamy segment pamięci współdzielonej?

- jaki problem występuje przy zapisie do bufora cyklicznego?

Zad. 12.5

Przepisz program mread oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

int main() {

int shmid = shmget(45281, MAX\*sizeof(int), IPC\_CREAT | 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczenie segmentu pamieci wspoldzielonej ");

exit(1);

}

printf("mread : mem = %p\n\n", mem);

printf("Data received:");

int i;

for (i = 0; i < N; i++) printf(" %d", mem[i % MAX]);

printf("\n");

shmdt(mem);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*)0);

system("ipcs -m");

return 0;

}

- jaki jest adres dowiązania segmentu pamięci współdzielonej dla procesu mread ?

- co możemy powiedzieć o adresach logicznych segmentów shm dla obu programów?

- co możemy powiedzieć o adresach fizycznych segmentów shm dla obu programów?

- jaka komenda podaje informacje o segmentach pamięci współdzielonej?

- w jaki sposób usuwamy segment pamięci współdzielonej?

- w jakim przypadku proces mread odczyta same zera?

Zad. 12.6

Przepisz program shm oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/shm.h>

#define M 1000

#define N 5

#define MAX 5

int main() {

int shmid = shmget(IPC\_PRIVATE, MAX\*sizeof(int), IPC\_CREAT | 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

pid\_t pid = fork();

if (pid == 0) {

int \*mem = (int\*)shmat(shmid, NULL, 0);

int i;

for (i = 0; i < M; i++) mem[i % MAX] = i;

printf("Child finished writing to shared memory.\n");

shmdt(mem);

}

if (pid > 0) {

int \*mem = (int\*)shmat(shmid, NULL, 0);

printf("Parent read: ");

int i;

for (i = 0; i < N; i++) printf("%d ", mem[i % MAX]);

printf("\n");

shmdt(mem);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*)0);

}

exit(0);

}

- uruchom program kilka razy i omów wyniki

- w jakim przypadku proces shm odczyta same zera?

- czy procesy rodzic i dziecko wymagają synchronizacji?

Zad. 12.7

Napisz program shm2 eliminujący błędy programu shm. Program powinien poprawnie odczytać N ostatnio zapisanych elementów.

Laboratorium 13

Zad. 13.1

Przepisz program thread oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

void \*hello(void \*arg) {

int i;

for (i = 0; i < 5; i++) {

printf("watek, i = %d\n", i);

sleep(1);

}

return NULL;

}

int main() {

pthread\_t mythread;

if (pthread\_create(&mythread, NULL, hello, NULL)) {

printf("Blad przy tworzeniu watku.\n");

abort();

}

printf("Proces czeka na zakonczenie watku.\n");

if (pthread\_join(mythread, NULL)) {

printf("Blad przy konczeniu watku.\n");

exit(1);

}

exit(0);

}

Zad. 13.2

Przepisz program thread2 oraz przeanalizuj jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int sum; // this data is shared by the thread(s)

void \*runner(void \*param); // threads call this function

int main(int argc, char \*argv[])

{

pthread\_t tid; // the thread identifier

pthread\_attr\_t attr; // set of thread attributes

pthread\_attr\_init(&attr); // set the default attributes of the thread

pthread\_create(&tid, &attr, runner, argv[1]); // create the thread

pthread\_join(tid, NULL); // wait for the thread to exit

printf("sum = %d\n", sum);

}

void \*runner(void \*param) // The thread will execute in this function

{

int acc = 0;

int n = atoi(param);

int i;

for (i = 1; i <= n; i++) acc = acc + i;

sum = acc;

pthread\_exit(0);

}

Zad. 13.3

Przepisz program sync ilustrujący dwa wątki, które inkrementują zmienną współdzieloną.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define TIMES 50

int x = 0;

void \*inc(void \*arg) {

int i;

for (i = 0; i < TIMES; i++) {

int reg = x;

reg = reg + 1;

x = reg;

}

return NULL;

}

int main() {

pthread\_t thread2;

if ( pthread\_create(&thread2, NULL, inc, NULL) ) {

printf("Blad przy tworzeniu watku.\n");

abort();

}

int i;

for (i = 0; i < TIMES; i++) {

int reg = x;

reg = reg + 1;

x = reg;

}

if ( pthread\_join(thread2, NULL) ) {

printf("Blad przy konczeniu watku.\n");

abort();

}

printf("x = %d\n", x);

exit(0);

}

- jakie zasoby współdzielą wątki?

- jakich zasobów nie współdzielą wątki?

- jak nazywa się zmienna współdzielona?

- w jaki sposób inkrementowana jest zmienna współdzielona?

- ile razy inkrementuje zmienną współdzieloną wątek główny?

- ile razy inkrementuje zmienną współdzieloną wątek poboczny?

- ile razy jest inkrementowana zmienna współdzielona?

- czy program daje poprawny wynik?

- czy program jest napisany poprawnie?

Zad. 13.4

Omów poniższy schemat ilustrujący możliwe błędy przy inkrementacji zmiennej współdzielonej.

- Scenario 1

: x = 0

\* Thread 0 \* Thread 1

reg = x ; reg = 0

reg = reg + 1 ; reg = 1

x = reg ; x = 1

reg = x ; reg = 1

reg = reg + 1 ; reg = 2

x = reg ; x = 2

: x = 2

- Scenario 2

: x = 0

\* Thread 0 \* Thread 1

reg = x ; reg = 0

reg = reg + 1 ; reg = 1 reg = x ; reg = 0

x = reg ; x = 1 reg = reg + 1 ; reg = 1

x = reg ; x = 1

: x = 1

- który scenariusz zachodzi w programie sync ?

Zad. 13.5

Napisz program sync2 ilustrujący poprawną inkrementację zmiennej współdzielonej przez dwa wątki.

- czy poprawiony program będzie czerpał korzyści z architektury wielordzeniowej?

Zad. 13.6

Napisz program sync3 na podstawie sync ilustrujący dwa wątki, które inkrementują zmienną współdzieloną przy pomocy instrukcji inkrementacji.

- czy teraz program będzie działał poprawnie?

Zad. 13.7

Napisz program sync4 na podstawie sync3 ilustrujący dwa wątki, które poprawnie inkrementują zmienną współdzieloną przy pomocy instrukcji inkrementacji.

Zad. 13.8

Napisz program sync5 na podstawie sync3 ilustrujący dwa wątki, które poprawnie inkrementują zmienną współdzieloną z wykorzystaniem mechanizmu zamków.

- który program wykona się szybciej sync5 czy sync4 ?

- zmierz czas wykonania programu sync5 i sync4

- czy oczekiwanie na zakończenie wątku pobocznego jest konieczne?

Zad. 13.9

Napisz program sync6 ilustrujący dwa wątki, które poprawnie inkrementują zmienną współdzieloną z wykorzystaniem mechanizmu spinlocków.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define TIMES 50000

pthread\_spinlock\_t lock;

int x = 0;

void \*inc(void \*arg) {

int i;

for (i = 0; i < TIMES; i++) {

pthread\_spin\_lock(&lock);

x++;

pthread\_spin\_unlock(&lock);

}

return NULL;

}

int main() {

if (pthread\_spin\_init(&lock, PTHREAD\_PROCESS\_PRIVATE)) {

fprintf(stderr, "Error initializing spinlock\n");

return 1;

}

pthread\_t thread2;

if ( pthread\_create(&thread2, NULL, inc, NULL) ) {

printf("Blad przy tworzeniu watku.\n");

abort();

}

int i;

for (i = 0; i < TIMES; i++) {

pthread\_spin\_lock(&lock);

x++;

pthread\_spin\_unlock(&lock);

}

if ( pthread\_join(thread2, NULL) ) {

printf("Blad przy konczeniu watku.\n");

abort();

}

pthread\_spin\_destroy(&lock);

printf("x = %d\n", x);

exit(0);

}

- porównaj czas wykonania programów sync6 i sync5

Zad. 13.11

Napisz program sync7 ilustrujący, ile razy wątek główny zdoła wykonać operację inkrementacji zanim wystartuje wątek poboczny.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int started = 0;

int x = 0;

void \*inc(void \*arg) {

started = 1;

return NULL;

}

int main() {

pthread\_t thread2;

if ( pthread\_create(&thread2, NULL, inc, NULL) ) {

printf("Blad przy tworzeniu watku.\n");

abort();

}

while (!started) {

int reg = x;

reg = reg + 1;

x = reg;

}

printf("x = %d\n", x);

exit(0);

}

- co oznacza wynik x = 0 ?

- gdzie w programie mamy mechanizm podobny do spinlocka?

Laboratorium 14

Zad. 14.1

1. Przepisz kod sem ilustrujący operacje na semaforach i wyjaśnij jego działanie.

#include <stdio.h>

#include <stdlib.h>

#include <sys/sem.h>

static struct sembuf sops;

void podnies(int semid, int semnum) {

sops.sem\_num = semnum;

sops.sem\_op = 1;

sops.sem\_flg = 0;

if (semop(semid, &sops, 1) == -1) {

perror("Podnoszenie semafora ");

exit(1);

}

}

void opusc(int semid, int semnum) {

sops.sem\_num = semnum;

sops.sem\_op = -1;

sops.sem\_flg = 0;

if (semop(semid, &sops, 1) == -1) {

perror("Opuszczanie semafora ");

exit(1);

}

}

2. Przepisz program producer ilustrujący poprawny zapis do bufora cyklicznego.

#include <stdio.h>

#include <stdlib.h>

#include <sys/sem.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

void podnies(int semid, int semnum);

void opusc(int semid, int semnum);

int main() {

int semid = semget(45281, 2, IPC\_CREAT | 0600);

if (semid == -1) {

perror("Tworzenie tablicy semaforow ");

exit(1);

}

if (semctl(semid, 0, SETVAL, (int)MAX) == -1) {

perror("Nadanie wartosci semaforowi 0 ");

exit(1);

}

if (semctl(semid, 1, SETVAL, (int)0) == -1) {

perror("Nadanie wartosci semaforowi 1 ");

exit(1);

}

int shmid = shmget(45281, MAX\*sizeof(int), IPC\_CREAT | 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczanie segmentu pamieci wspoldzielonej ");

exit(1);

}

printf("producer : mem = %p\n\n", mem);

int i;

for (i = 0; i < N; i++) {

opusc(semid, 0); // 0 wolne

mem[i % MAX] = i;

podnies(semid, 1); // 1 zajete

}

printf("Data written to shared memory.\n");

shmdt(mem);

return 0;

}

- sprawdź na jakiej funkcji jądra usypiany jest proces producer

3. Przepisz program consumer ilustrujący poprawny odczyt z bufora cyklicznego.

#include <stdio.h>

#include <stdlib.h>

#include <sys/sem.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

void podnies(int semid, int semnum);

void opusc(int semid, int semnum);

int main() {

int semid = semget(45281, 2, 0600);

if (semid == -1) {

perror("Uzyskanie id tablicy semaforow ");

exit(1);

}

int shmid = shmget(45281, MAX\*sizeof(int), 0600);

if (shmid == -1) {

perror("Uzyskanie id segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczenie segmentu pamieci wspoldzielonej ");

exit(1);

}

printf("consumer : mem = %p\n\n", mem);

printf("Data received:");

int i;

for (i = 0; i < N; i++) {

opusc(semid, 1); // 1 zajete

printf(" %d", mem[i % MAX]);

podnies(semid, 0); // 0 wolne

}

printf("\n");

shmdt(mem);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*)0);

semctl(semid, 0, IPC\_RMID);

system("ipcs -ms");

return 0;

}

- który program należy uruchomić jako pierwszy?

Zad. 14.2

Napisz program prod-cons realizujący zad. 14.1 w jednym programie przy pomocy procesów.

Zad. 14.3

Napisz program prod-cons2 realizujący zad. 14.1 w jednym programie przy pomocy wątków. Dodatkowo, wątek konsumenta zwraca na stercie dane odebrane od producenta, a funkcja main wypisuje te dane.

Zad. 14.4

1. Przepisz program producers ilustrujący producentów zapisujących do bufora cyklicznego.

#include <stdio.h>

#include <stdlib.h>

#include <sys/sem.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

#define indexZ mem[MAX]

void podnies(int semid, int semnum);

void opusc(int semid, int semnum);

int main() {

int shmid = shmget(45281, (MAX+2)\*sizeof(int), IPC\_CREAT | 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczanie segmentu pamieci wspoldzielonej ");

exit(1);

}

int semid = semget(45281, 4, IPC\_CREAT | IPC\_EXCL | 0600);

if (semid == -1) {

semid = semget(45281, 4, 0600);

if (semid == -1) {

perror("Tworzenie tablicy semaforow ");

exit(1);

}

} else {

if (semctl(semid, 0, SETVAL, (int)MAX) == -1) {

perror("Nadanie wartosci semaforowi 0 ");

exit(1);

}

if (semctl(semid, 1, SETVAL, (int)0) == -1) {

perror("Nadanie wartosci semaforowi 1 ");

exit(1);

}

if (semctl(semid, 2, SETVAL, (int)1) == -1) {

perror("Nadanie wartosci semaforowi 2 ");

exit(1);

}

if (semctl(semid, 3, SETVAL, (int)1) == -1) {

perror("Nadanie wartosci semaforowi 3 ");

exit(1);

}

}

printf("producer : mem = %p\n\n", mem);

indexZ = 0;

int i;

for (i = 0; i < N; i++) {

opusc(semid, 0); // 0 wolne

opusc(semid, 2); // 2 producent

mem[indexZ] = i;

indexZ = (indexZ + 1) % MAX;

podnies(semid, 2); // 2 producent

podnies(semid, 1); // 1 zajete

}

printf("Data written to shared memory.\n");

shmdt(mem);

return 0;

}

2. Przepisz program consumers ilustrujący konsumentów odczytujących z bufora cyklicznego.

#include <stdio.h>

#include <stdlib.h>

#include <sys/sem.h>

#include <sys/shm.h>

#define N 10

#define MAX 5

#define indexO mem[MAX+1]

void podnies(int semid, int semnum);

void opusc(int semid, int semnum);

int main() {

int shmid = shmget(45281, (MAX+2)\*sizeof(int), 0600);

if (shmid == -1) {

perror("Tworzenie segmentu pamieci wspoldzielonej ");

exit(1);

}

int \*mem = (int\*)shmat(shmid, NULL, 0);

if (mem == (void\*)-1) {

perror("Przylaczanie segmentu pamieci wspoldzielonej ");

exit(1);

}

int semid = semget(45281, 4, IPC\_CREAT | IPC\_EXCL | 0600);

if (semid == -1) {

semid = semget(45281, 4, 0600);

if (semid == -1) {

perror("Tworzenie tablicy semaforow ");

exit(1);

}

} else {

if (semctl(semid, 0, SETVAL, (int)MAX) == -1) {

perror("Nadanie wartosci semaforowi 0 ");

exit(1);

}

if (semctl(semid, 1, SETVAL, (int)0) == -1) {

perror("Nadanie wartosci semaforowi 1 ");

exit(1);

}

if (semctl(semid, 2, SETVAL, (int)1) == -1) {

perror("Nadanie wartosci semaforowi 2 ");

exit(1);

}

if (semctl(semid, 3, SETVAL, (int)1) == -1) {

perror("Nadanie wartosci semaforowi 3 ");

exit(1);

}

}

printf("consumer : mem = %p\n\n", mem);

printf("Data received:");

indexO = 0;

int i;

for (i = 0; i < N; i++) {

opusc(semid, 1); // 1 zajete

opusc(semid, 3); // 3 konsument

printf(" %d", mem[indexO % MAX]);

indexO = (indexO + 1) % MAX;

podnies(semid, 3); // 3 konsument

podnies(semid, 0); // 0 wolne

}

printf("\n");

shmdt(mem);

return 0;

}

- kilka razy uruchom kolejno 2 producentów i 2 konsumentów i omów wyniki

- po wykonaniu testów usuń semafor i segment pamięci współdzielonej

Zad. 14.5

Napisz program thread-sum ilustrujący wątek pobierający tablicę liczb i zwracający ich sumę.

Zad. 14.6 \*

Napisz program thread-calc ilustrujący dwa wątki. Pierwszy pobiera tablicę liczb i zwraca minimum i maksimum z tej tablicy, a drugi zwraca średnią elementów z tablicy.

Zad. 14.7

Napisz program prod-cons3 realizujący zad. 14.1 w jednym programie przy pomocy zmiennych warunkowych.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define N 10

#define MAX 5

int buffer[MAX];

int count = 0; // Number of items in the buffer

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t spaceAvailable = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t datumAvailable = PTHREAD\_COND\_INITIALIZER;

void \*producer(void \*arg) {

for (int i = 0; i < N; ++i) {

pthread\_mutex\_lock(&mutex);

while (count == MAX) {

// Buffer is full, wait for a consumer to signal that there is space

pthread\_cond\_wait(&spaceAvailable, &mutex);

}

// Produce an item and add it to the buffer

buffer[count++] = i;

// Signal to consumer that there is a new item in the buffer

pthread\_cond\_signal(&datumAvailable);

pthread\_mutex\_unlock(&mutex);

}

pthread\_exit(NULL);

}

void \*consumer(void \*arg) {

printf("consumer received:");

for (int i = 0; i < N; ++i) {

pthread\_mutex\_lock(&mutex);

while (count == 0) {

// Buffer is empty, wait for a producer to signal that there is an item

pthread\_cond\_wait(&datumAvailable, &mutex);

}

printf(" %d", buffer[--count]);

// Signal to producer that there is space in the buffer

pthread\_cond\_signal(&spaceAvailable);

pthread\_mutex\_unlock(&mutex);

}

printf("\n");

pthread\_exit(NULL);

}

int main() {

pthread\_t producer\_thread;

pthread\_t consumer\_thread;

pthread\_create(&producer\_thread, NULL, producer, NULL);

pthread\_create(&consumer\_thread, NULL, consumer, NULL);

pthread\_join(producer\_thread, NULL);

pthread\_join(consumer\_thread, NULL);

return 0;

}

- czy wszystkie wyprodukowane elementy zostały odebrane przez konsumenta?

- jaka jest kolejność odbierania wyprodukowanych elementów?

Zad. 14.8

Napisz program prod-cons4 realizujący zad. 14.1 w jednym programie przy pomocy zmiennych warunkowych i bufora cyklicznego.

Zad. 14.9

Napisz program broadcast ilustrujący zawiadomienie wszystkich oczekujących wątków, gdy spełniony jest określony warunek.

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#define N 5

int predicate = 0;

pthread\_cond\_t cond = PTHREAD\_COND\_INITIALIZER;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

void \*waiting(void \*arg) {

pthread\_mutex\_lock(&mutex);

while (!predicate)

pthread\_cond\_wait(&cond, &mutex);

pthread\_mutex\_unlock(&mutex);

printf("Thread %ld has terminated.\n", (long)arg);

pthread\_exit(NULL);

}

void \*broadcasting(void \*arg) {

pthread\_mutex\_lock(&mutex);

printf("I will signal all waiting threads in 3 seconds.\n\n");

sleep(3);

predicate = 1;

pthread\_cond\_broadcast(&cond);

pthread\_mutex\_unlock(&mutex);

pthread\_exit(NULL);

}

int main() {

pthread\_t thread[N+1];

long i;

for (i = 0; i < N; i++) {

pthread\_create(&thread[i], NULL, waiting, (void\*)i);

}

pthread\_create(&thread[N], NULL, broadcasting, NULL);

for (i = 0; i <= N; i++) {

pthread\_join(thread[i], NULL);

}

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&cond);

return 0;

}

Zad. 14.10 \*

Napisz program readers-writers ilustrujący problem czytelników i pisarzy, gdzie wielu czytelników może równolegle czytać dane, ale tylko jeden pisarz może je zapisywać. Użyj zmiennych warunkowych i zamków do synchronizacji dostępu.

Zad. 14.11 \*

Napisz program bariera ilustrujący koordynację współbieżnych wątków. Wątki czekają przed barierą, a po jej pokonaniu, wszystkie jedocześnie wznawiają wykonanie.

Zad. 14.12 \*

Napisz program semaphor ilustrujący implementację semafora przy pomocy zmiennych warunkowych i zamków.

Zad. 14.13

Napisz program threads-sum sumujący tablicę N początkowych liczb naturalnych sekwencyjnie oraz za pomocą dwóch wątków korzystających z jednej funkcji add. Do funkcji należy przekazać przedział sumowania. Przykładowa sesja dla N = 10:

sum1 = 51

sum2 = 51

left[2] = {0, 4}

right[2] = {5, 9}

Zad. 14.14 \*

Napisz program threads-sum2 analogicznie do powyższego, który dodatkowo wyświetla czas wykonania sumy sekwencyjnej i współbieżnej. Przykładowa sesja dla tablicy 10MB.

sum1 = 1336807840

sum2 = 1336807840

sum1 time = 0.017000

sum2 time = 0.008000

Zad. 14.15 \*

Napisz program min-max wyznaczający minimum i maksimum 10MB tablicy liczb losowych z przedziału [-(RAND\_MAX+1)/2, RAND\_MAX/2] sekwencyjnie oraz za pomocą dwóch wątków korzystających z jednej funkcji min\_max, która zwraca jednocześnie minimum i maksimum. Program dodatkowo wyświetla czas wykonania obliczeń sekwencyjnych i współbieżnych. Przykładowa sesja:

[-(RAND\_MAX+1)/2, RAND\_MAX/2] = [-1073741824, 1073741823]

min1 = -1073741769

max1 = 1073741655

min2 = -1073741769

max2 = 1073741655

min1 max1 time = 0.037242

min2 max2 time = 0.019304

Suplement

Zad.

Napisz program scheduler.c ilustrujący różne algorytmy szeregowania procesów. Program implementuje funkcje:

void burstTime(struct Process p[], int size);

- wypisuje czas wykonania procesów

void waitingTime(struct Process p[], int size);

- wypisuje czas oczekiwania procesów na przydział procesora

double avgWaitingTime(struct Process p[], int size);

- oblicza średni czas oczekiwania procesów na przydział procesora

void ganttChart(struct Process p[], int size);

- wypisuje diagram Gantta \*

void applySJF(struct Process p[], int size);

- szereguje procesy według algorytmu SJF (ang. Shortest Job First)

Metody