Europejska Wyższa Szkoła Informatyczno-Ekonomiczna
w Warszawie

Przedmiot: Podstawy programowania - „Projekt zaliczeniowy”

Temat:

Obliczenia matematyczne
Wykonywanie obliczeń w osobnych wątkach

Prowadził: mgr inż. R. Berliński

Opracował: J. Zdybel

Spis treści

1. Wstęp

Program forks, ma na celu zaprezentowanie działania funkcji fork() oraz wskaźników do zmiennych. Ogólny schemat działania polega na wprowadzeniu dwóch zmiennych typu int i wykonanie obliczeń i ich wyświetlenie w dwóch osobnych procesach.

1. Opis projektu

Program „forks” napisany został w języku C wg standardu ANSI C.
Program skompilowano kompilatorem gcc w wersji 4.9.2 na platformie Debian 8.2

Szczegółowy opis struktury programu

forks.c

Kod pliku i opis

Program rozpoczynam od deklaracji nagłówków. Standardowo zadeklarowane są dwie podstawowe biblioteki

```
#include <stdio.h>
#include <stdlib.h>
```

stdio.h jest biblioteką umożliwiającą wykonywanie podstawowych operacji wejścia /wyjścia
stdlib.h jest biblioteką standardową, zawierającą standardowe narzędzia języka C.

```
int  iloczyn,  a,  b;
```
Deklaracja dwóch podstawowych zmiennych typu int

```
int * w_iloczyn, * w_a, * w_b;
```
Deklaracja wskaźników do zmiennych zadeklarowanych wcześniej

```
float iloraz, suma;
```
Deklaracja dwóch podstawowych zmiennych typu float

```
float *w_iloraz, *w_suma;
```
Deklaracja wskaźników do zmiennych zadeklarowanych wcześniej

Poniższy blok służy do pobrania podstawowych wartości koniecznych do wykonania programu:
W kolejności, program pyta o podanie wartości elementu 1, po czym wprowadzoną wartość dodaje do zmiennej a. W kroku drugim dodaje wartość zmiennej b

```
printf("Wprowadź wartrość argumentu 1: ");
```

```
scanf("%d", & a);
printf("Wprowadź wartrość argumentu 2: ");
scanf("%d", & b);
```

Przekazanie wskaźników do zmiennych następuje w linii 22 i 23:

```
w_a = & a;
w_b = & b;
```

Do zadeklarowanego wcześniej wskaźnika np. *w_a, przekazywany jest adres komórki w którym przechowywana jest zmienna za pomocą operatora &, tutaj &a.

W linii 29 następuje utworzenie nowego procesu

```
pid_t  id =  fork();
```

Funkcja fork(), tworzy kopię procesu. Utworzony proces potomny jest niemal identyczną kopią procesu macierzystego. Proces potomny dziedziczy kopię kodu, danych, stosu oraz otwartych deskryptorów plików i sygnałów rodzica. Powstały proces potomny różni się identyfikatorem procesu.

W przypadku gdy funkcja fork() zakończy się powodzeniem, zwraca PID procesu potomnego do rodzica, a do procesu potomnego zwraca 0. Przypadek gdy zwracana wartość do procesu rodzica wynosi -1 oznacza że nie utworzono procesu potomnego.

Powyższą właściwość funkcji fork() wykorzystuję w liniach 34 i 45 gdzie warunkując powstały identyfikator procesów dzielę wykonywanie obliczeń między proces macierzysty i potomny.
W przypadku gdy rezultat wykonania funkcji fork() równy jest zero (pid procesu wynosi 0) znajdujemy się w procesie potomnym (linia 35 – 44).

```
if ( id == 0) {
printf("Id procesu: %d\n",  id);
iloczyn = (* w_a) * (* w_b);
w_iloczyn = & iloczyn;
printf("Iloczyn liczb: %d * %d = %d\n", * w_a, * w_b, *
w_iloczyn);
printf("\n\n --- Koniec procesu potomnego --- \n\n");
}
```

W procesie potomnym następuje operacja mnożenia wprowadzonych na początku programu wartości. Operacja ta wykonywana jest na wskaźnikach zadeklarowanych na początku programu. Wartości przechowywane w komórkach pamięci pobierane są za pomocą wskaźników *w_a i *w_b Po wykonaniu operacji mnożenia, do wskaźnika w_iloczyn przekazywana jest informacja o adresie pamięci w której przechowywana jest wartość zmiennej.

```
w_iloczyn = & iloczyn
```

W sytuacji gdy id > 0 znajdujemy się w procesie macierzystym (linia 46 do 67)

```
printf("\n\n --- Początek procesu macierzystego --- \n\n");
printf("Mam zadeklarowane dwie zmienne: \na: %i, \nb: %i\n", *
```

```
w_a, * w_b);
printf("czekam aż proces potomny się skończy... \n");
```

W celu zachowania kolejności obliczeń w linii 60 czekamy na zakończenie procesu potomnego, w którym program wykonuje operację mnożenia, i dopiero po zakończeniu procesu potomnego następuje wykonanie operacji dzielenia w linii 64.

```
int  status = 255;
id =  wait(& status);
```

Za pomocą funkcji wait() zawieszamy proces wywołujący (macierzysty) do czasu zakończenia pracy przez proces potomny. Pomyślne wywołanie wait() zwraca identyfikator zakończonego procesu potomnego.
W przypadku gdy proces wywoła wait() i nie znajdzie potomków, wait() zwraca -1.

```
id =  waitpid( id, & status, 0);
```

Tutaj również cała operacja odbywa się na wskaźnikach i wartościach do których wskaźniki te się odwołują.

```
iloraz = (float) * w_a / (float) * w_b;
w_iloraz = & iloraz;
}
```

2. Kompilacja aplikacji

Kompilacja odbywa się za pomoca polecenia

```
gcc -o forks forks.c
```

Aplikacja kompilowana jest jak widać z czterech plików .c oraz pliku nagłówkowego main.h, dzięki czemu proces kompilacji łączy wszystkie pliki w jeden plik wykonywalny „spis_telefonow” któremu dodatkowo powinniśmy nadać uprawnienia do wykonywania za pomocą polecenia

```
chmod +x forks
```

Teraz, z poziomu katalogu aplikacji możemy uruchomić program poleceniem

```
./forks
```

Przykładowe wyjście
Program wykonujący działania mnożenia i dzielenia w dwóch osobnych procesach

Po wyjściu z procesu potomnego, wykonywana jest operacja sumowania rezultatów mnożenia i dzielenia

Wprowadź wartrość argumentu 1: 4

Wprowadź wartrość argumentu 2: 3

Parametr a po wskazniku w_a: 4

Parametr b po wskazniku w_b: 3

--- Początek procesu macierzystego ---

Mam zadeklarowane dwie zmienne:

a: 4,

b: 3

czekam aż proces potomny się skończy...

--- Początek procesu potomnego ---

Id procesu: 0

Iloczyn liczb: 4 * 3 = 12

--- Koniec procesu potomnego ---

--- Wyjście z procesu potomnego, pozostałe operacje ---

12

1.333333

Suma liczb: 12 + 1.333333 = 13.333333

id1: 5555

id2: -1

Id procesu: -1

Iloraz liczb: 4 / 3 = 1.333333

--- Koniec procesu macierzystego ---

--- Wyjście z procesu potomnego, pozostałe operacje ---

12

1.333333

Suma liczb: 12 + 1.333333 = 13.333333

3. Bibliografia

1. mgr inż. R. Berliński: *Ćwiczenia: Systemy Operacyjne*
2. Stephen Prata : *Język C. Szkoła programowania*, – Wydanie V
3. Graham Glass, King Ables: *Linux dla programistóœ i użytkowników*
4. Wikipedia: *C_Programming*
5. strefakursow.pl - *Szkoła programowania w języku C*

4. Kod źródłowy programu