USUX

Laboratorium 8

Narzędzia programistyczne

Oświadczam, że niniejsza praca stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Użytkowanie systemu UNIX została wykonana przeze mnie samodzielnie.

Arkadiusz Sowa
300506

1.Wykorzystując kompilator gcc wykonać następujące czynności:

1.1 Przenieść prywatne pliki nagłówkowe (*.h) do innego katalogu niż pliki źródłowe *.c,

Przenoszenie wszystkich plików .h można wykonać w prosty sposób za pomocą komend.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % mv *.h pliki_h
```

1.2. Skompilować wszystkie pliki źródłowe i utworzyć z nich program wynikowy o nazwie prog.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc delta.c pierw.c rkw.c -Ipliki_h -o sowa
/usr/bin/ld: /tmp/ccalcZkP.o: in function `Pierw':
pierw.c:(.text+0x44): undefined reference to `sqrt'
/usr/bin/ld: pierw.c:(.text+0x78): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```

Otrzymałem error, iż nie zostało rozpoznane ‘sqrt’ więc dołączyłem bibliotekę matematyczna poleceniem -lm

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc delta.c pierw.c rkw.c -Ipliki_h -lm -o prog
```

W katalogu p2 powstał plik ‘prog’, wcześniej używałem nazwy ‘sowa’ ale doczytałem w instrukcji że nazwa jest już z góry narzucona, dlatego dokonałem zmiany

1.3 Porównać rozmiar pliku wynikowego otrzymanego po kompilacji z włączoną i wyłączoną optymalizacją.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -O2 delta.c pierw.c rkw.c -Ipliki_h -lm -o prog_O2
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -O3 delta.c pierw.c rkw.c -Ipliki_h -lm -o prog_O3
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc delta.c pierw.c rkw.c -Ipliki_h -lm -o prog
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -O4 delta.c pierw.c rkw.c -Ipliki_h -lm -o prog_O4
```

Znalazłem informację iż maksymalny poziom optymalizacji to 3, jednakże O4 równie zadziałoło n, nie widać jednak różnicy w wielkości plików pomiędzy O2, O3 i O4.

```
-rwxr-xr-x 1 arek arek 17048 gru  3 13:31 prog
-rwxr-xr-x 1 arek arek 17056 gru  3 13:30 prog_O2
-rwxr-xr-x 1 arek arek 17056 gru  3 13:31 prog_O3
-rwxr-xr-x 1 arek arek 17056 gru  3 13:31 prog_O4
```

Możemy zauważyć iż pliki z włączoną optymalizacją zajmują więcej miejsca na dysku.

1.4. Znaleźć w kodzie źródłowym makro sterujące procesem prekompilacji i wykorzystując odpowiednią opcję programu gcc wykonać punkt 1b w dwóch wersjach.

```
#ifdef ZESPOLONE
double* PierwZesp(double a, double b, double delta, int flaga)
{
```

ZESPOLONE jest makro sterującym procesem prekompilacji.

Wykonałem kompilacje dla funkcji -D (define) i -U (undefine)

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -DZESPOLONE delta.c pierw.c rkw.c -Ipliki_h -lm -o prog_d_zesp
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -UZESPOLONE delta.c pierw.c rkw.c -Ipliki_h -lm -o prog_u_zesp
```

Możemy zauważyć, iż domyślna kompilacja bez użycia makro sterującym procesem prekompilacji ma zajmuje identyczna ilość pamięci oraz ma taką samą funkcjonalność jak kompilacja z funkcją -U.

Funkcja -D pozwala nam na dodanie funkcjonalności do programu.

```
-rwxr-xr-x 1 arek arek 17048 gru  3 13:31 prog
-rwxr-xr-x 1 arek arek 17128 gru  3 13:46 prog_d_zesp
-rwxr-xr-x 1 arek arek 17048 gru  3 13:46 prog_u_zesp
```

2. Posługując się programem ar wykonać operacje:

2.1 Zbudować własną bibliotekę statyczną libusux.a z wybranych plików obiektowych.

Kompilowanie plików .c, -c pozwala na utworzenie plików .o

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc pierw.c delta.c -Ipliki_h -c
```

Generowanie biblioteki.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % ar cr libusux.a delta.o pierw.o
```

Otrzymana biblioteka

rkw. nie zostało dodane do biblioteki ponieważ jest to plik zawierający main daltegoo nie ma sensu dodawanie jej do biblioteki.

2.2 Wykorzystać stworzoną bibliotekę do utworzenia tego samego programu co w punkcie 1b.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc rkw.c libusux.a -Ipliki_h -lm -o prog_lib
```

Pliki delta.c oraz pierw.c zostają teraz zastąpione przez naszą bibliotekę.

3. Posługując się programem gcc wykonać operacje.

3.1 Zbudować własną bibliotekę dzieloną (dynamiczną) libusux.so z wybranych plików obiektowych.

-fpic - generuje kod typu PIC (ang. *Position-Independent Code*) wymagany do utworzenia biblioteki dzielonej (dynamicznie linkowanej),
-fPIC

-shared - tworzy obiekt współdzielony (ang. *shared*), który może być później skonsolidowany z innymi obiektami w celu utworzenia pliku wykonywalnego,

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc pierw.c delta.c -Ipliki_h -fPIC -c
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -shared pierw.o delta.o -o libusux.so
```

3.2 Wykorzystać stworzoną bibliotekę do utworzenia tego samego programu co w punkcie 1b.

-L. ścieżka do biblioteki

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc rkw.c -L. -Ipliki_h -lusux -lm -o  prog_lib_dyn
```

3.3 Zmodyfikować ścieżkę poszukiwań bibliotek, aby umożliwić wykonywanie programu.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % export LD_LIBRARY_PATH=$LC_LIBRARY_PATH:.
```

4. Obejrzeć tablicę symboli programu prog i biblioteki libusux.a . Usunąć tablicę symboli z obu tych plików. Sprawdzić działanie programu po usunięciu tablicy symboli. Powtórzyć punkt 2b i wytłumaczyć ewentualne rożnice w działaniu kompilatora.

Obejrzeć tablicę symboli programu prog i biblioteki libusux.a . Usunąć tablicę symboli z obu tych plików.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % nm prog
0000000000004010 B __bss_start
0000000000004010 b completed.8055
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000004000 D __data_start
```

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % strip prog
```

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % nm libusux.a

delta.o:
0000000000000000 T Delta

pierw.o:
                 U _GLOBAL_OFFSET_TABLE_
0000000000000000 T Pierw
                 U sqrt
```

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % strip libusux.a
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % nm libusux.a

delta.o:
nm: delta.o: no symbols

pierw.o:
nm: pierw.o: no symbols
```

Powtórzyć punkt 2b i wytłumaczyć ewentualne rożnice w działaniu kompilatora.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc rkw.c libusux.a -Ipliki_h -lm -o prog_lib_zad4
/usr/bin/ld: libusux.a: error adding symbols: archive has no index; run ranlib t
o add one
collect2: error: ld returned 1 exit status
```

Dostajemy błąd mówiący o braku indeksów, uniemożliwiający stworzenie programu.

5. Używając programów gcc i gdb wykonać następujące polecenia:

5.1. Utworzyć program prog w taki sposób by umożliwić śledzenie jego pracy za pomocą gdb.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gcc -g rkw.c delta.c pierw.c -Ipliki_h -lm -o prog_g
```

Funkcja -g umożliwia śledzenie pracy programu za pomocą gdb.

5.2. Obejrzeć kod źródłowy przy pomocy gdb, wybrać miejsca dla kilku pułapek i je ustawić.

Otwieramy program za pomocą gdb

Na oglądnie kodu źródłowego pozwala funkcja ‘list’ wtświetlająca 10 lini kodu, kolejne linie możemy zobaczyć po naciśnięciu ‘enter’.

```
(gdb) list
1       #include "rkw.h"
2
3       void main(int argc, char* argv[])
4       {
5               double a, b, c, delta;
6               double x1, x2;
7       #ifdef ZESPOLONE
8               double *x1z, *x2z;
9       #endif
10
```

Pułapki ustawiamy za pomocą ‘b rkw.c:2’ rkw.c to nazwa pliku, a 2 to linia kodu, w której ustawiamy pułapkę. Dla linii 2, pułapka zostanie ustawiona w naszym przypadku na w linii 4, ponieważ to pierwsza sensowna linia kodu, w której można ustawić pułapkę

```
(gdb) b rkw.c:10
Breakpoint 3 at 0x120b: file rkw.c, line 11.
(gdb) b rkw.c:2
Note: breakpoints 1 and 2 also set at pc 0x11e9.
Breakpoint 4 at 0x11e9: file rkw.c, line 4.
(gdb) b rkw.c:16
Breakpoint 5 at 0x1236: file rkw.c, line 16.
(gdb) b rkw.c:20
Breakpoint 6 at 0x12c8: file rkw.c, line 20.
```

5.3. Używając odpowiedniego polecenia programu gdb podać argumenty wywołania programu.

```
arek@arek-VirtualBox ~/lab8/przyklad2/p2
 % gdb --args prog_g 1 2 1
```

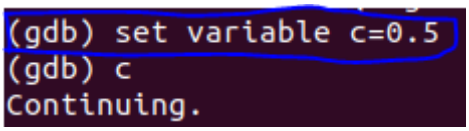
5.4. Ustawić tryb śledzenia dla wybranej zmiennej.

```
Breakpoint 1, main (argc=4, argv=0x7fffffffdf28) at rkw.c:16
16              sscanf(argv[1], "%lf", &a);
(gdb) watch a
Hardware watchpoint 4: a
(gdb) c
Continuing.

Hardware watchpoint 4: a

Old value = 4.6355705385483528e-310
New value = 1
0x00007ffff7ce4ca7 in __vfscanf_internal (s=s@entry=0x7fffffffdc20,
    format=format@entry=0x555555556025 "%lf",
    argptr=argptr@entry=0x7fffffffdc08, mode_flags=mode_flags@entry=2)
    at vfscanf-internal.c:2450
2450    vfscanf-internal.c: No such file or directory.
(gdb) watch a
Hardware watchpoint 5: a
```

5.5. Uruchomić program.

```
The program is not being run.
(gdb) r
Starting program: /home/arek/lab8/przyklad2/p2/prog_g 1 2 1
```

5.6. W trakcie krokowego wykonywania programu zmienić wartość zmiennej z punktu 5d.



Ustawienie zmiennej c na 0,5