Przemysław Szymoniak

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 1.

Celem laboratorium była organizacja środowiska tworzenia oprogramowania w systemie Linux oraz przeprowadzenie pomiarów czasu CPU, zegarowego wykonywania operacji przy użyciu interfejsu procedur pomiaru czasu.

W ramach zajęć zrealizowałem następujące kroki:

- skopiowałem wskazane pliki („**Makefile**”, „**pomiar_czasu.c**”, „**pomiar_czasu.h**”, „**moj_program.c**”) do katalogu roboczego „**lab_1**”
- zmodyfikowałem plik źródłowy „**moj_program.c**”:
  - ✓ **#include "pomiar_czasu.h"**

```
…

inicjuj_czas();
for(i=0;i<liczba;i++){
        printf("%d ",k+i);
}
printf("\n");
printf("Czas wykonania %d operacji wejścia / wyjścia: \n", liczba);
drukuj_czas();

…

t1=czas_zegara();
t2=czas_CPU();
for(i=0;i<liczba;i++){
        a = 1.000001*a+0.000001;
}
t1=czas_zegara()-t1;
t2=czas_CPU()-t2;
```

- zmodyfikowałem plik „**Makefile**” tak aby poprawnie tworzyć plik binarny „**moj_program**”:
  - ✓ **moj_program.o: moj_program.c pomiar_czasu.h**
    $(CCOMP) -c $(OPT) moj_program.c
  - ✓ kompilacje pliku źródłowego „**moj_program.c**” przez wykonanie w terminalu polecenia: **make**
- uruchomienie programu
- przetestowałem działanie uzyskują następujące wyniki (**najkrótsze czasy, najdłuższe czasy**):

Pomiary przy użyciu opcji optymalizacji w wersji do debugowania					
Lp.	**Czas wykonania 1000 operacji we/wy**			**Czas wykonania 1000 operacji arytmetycznych mnożenia**	
	Czas standard.	**Czas CPU**	**Czas zegarowy**	**Czas CPU**	**Czas zegarowy**
1.	**0.000000**	**0.004658**	**0.020673**	0.000289	0.000361
2.	0.000000	0.007469	0.051436	**0.000204**	**0.000350**
3.	0.000001	0.017540	0.056388	**0.000416**	**0.000539**
4.	0.000000	0.008295	0.042655	0.000247	0.000412
5.	**0.000001**	**0.018519**	**0.057720**	0.000382	0.000467

Pomiary przy użyciu opcji optymalizacji w wersji do mierzenia czasu					
Lp.	**Czas wykonania 1000 operacji we/wy**			**Czas wykonania 1000 operacji arytmetycznych mnożenia**	
	Czas standard.	**Czas CPU**	**Czas zegarowy**	**Czas CPU**	**Czas zegarowy**
1.	**0.000000**	**0.007107**	**0.021099**	0.000183	0.000206
2.	0.000001	0.010117	0.035946	**0.000154**	**0.000200**
3.	0.000001	0.014127	0.054166	**0.000235**	**0.000381**
4.	**0.000001**	**0.015783**	**0.066726**	0.000203	0.000280
5.	0.000001	0.011439	0.039744	0.000173	0.000205

W ramach zadań dodatkowych zrealizowałem:

- przeniosłem pliki związane z pomiarem czasu do osobnego katalogu „**pomiar_czasu**” a te związane z programem do katalogu „**moj_program**”
- utworzyłem bibliotekę „**libpomiar_czasu.a**” z pliku „**pomiar_czasu.o**” w katalogu „**pomiar_czasu**” za pomocą polecenia:
  - ✓ **ar -rs libpomiar_czasu.a pomiar_czasu.o**
- Zmieniłem plik „**Makefile**” tak aby zamiast z plików źródłowych i pośrednich pomiaru czasu korzystał ze stworzonej wcześniej biblioteki „**libpomiar_czasu.a**”:
  - ✓ usunąłem odniesienia do plików „**pomiar_czasu.c”** i „**pomiar_czasu.o**” oraz stworzyłem dla „**pomiar_czasu.o**” osobny „**Makefile**” w folderze „**pomiar_czasu**”
  - ✓ nadałem właściwe wartości symbolom **LIB** i **INC**
  - ✓ dodałem właściwe wykorzystanie symboli **LIB** i **INC** w trakcie kompilacji
  - ✓ Zmiany w pliku **Makefile**:

```
# pliki nagłówkowe
INC = -I../pomiar_czasu

# biblioteki
LIB = -L../pomiar_czasu –lm

# zależności i komendy
moj_program: moj_program.o
        $(LOADER) $(OPT) moj_program.o -o moj_program $(LIB)

# jak uzyskać plik moj_program.o ?
moj_program.o: moj_program.c ../pomiar_czasu/pomiar_czasu.h
        $(CCOMP) -c $(OPT) moj_program.c $(INC)
```

- wykonanie polecenia: **make clean && make** oraz sprawdzenie poprawności działania programu wraz z końcowym pomiarem czasu pokazanym w tabelach w pierwszej części sprawozdania.

Wnioski:

- dzięki wykorzystaniu narzędzia make oraz odpowiednio modyfikowanego pliku **Makefile** udało się zrealizować sprawne i elastyczne tworzenie kodu binarnego z plików źródłowych
- pomiar czasu realizacji procedur wykazał, że:
  - ✓ czas realizacji procedur wejścia/wyjścia jest znacznie dłuższy niż czas realizacji operacji arytmetycznych
  - ✓ przy realizacji procedur wejścia/wyjścia czas wykorzystania procesora (czas CPU) stanowi bardzo mały procent czasu realizacji operacji (procesor zdecydowaną większość czasu czeka na wykonanie operacji przez twardy dysk)
- wykorzystanie optymalizacji w wersji do mierzenia czasu przyniosło oczekiwane rezultaty przy obliczaniu czasu dla 1000 operacji arytmetycznych mnożenia:
  - ✓ najkrótszy czas CPU zmniejszył o **5 x 10^{-5}** a czas zegarowy zaś **o 1.5x10^{-4}**, najdłuższy czas CPU zmniejszył się **1.81x10^{-4}** natomiast czas zegarowy o **1.58x10^{-4}** w odniesieniu do wersji optymalizowanej do debugowania
  - ✓ zmniejszenie rozbieżności pomiędzy poszczególnymi pomiarami na skutek zmniejszenia różnicy pomiarowej między czasem najdłuższym oraz najkrótszym
- wykorzystanie optymalizacji w wersji do mierzenia czasu przyniosło mizerne rezultaty przy obliczaniu czasu dla 1000 operacji we/wy
  - ✓ najkrótszy czas CPU zwiększył się o **2.5 x 10^{-3}** a czas zegarowy zaś **o 4.26x10^{-4}**, najdłuższy czas CPU zmniejszył się **2.74x10^{-3}** natomiast czas zegarowy zwiększył się o **9x10^{-3}** w odniesieniu do wersji optymalizowanej do debugowania
  - ✓ zmniejszenie rozbieżności pomiędzy poszczególnymi pomiarami dla czasu CPU na skutek spadku różnicy między maksymalnym i minimalnym pomiarem o wartość **5.2 x 10^{-3}**
  - ✓ zwiększenie rozbieżności pomiędzy poszczególnymi pomiarami dla czasu zegarowego na skutek wzrostu różnicy między maksymalnym i minimalnym pomiarem o wartość **8.6 x 10^{-3}**
- wykorzystanie optymalizacji do pomiaru czasu w każdym przypadku skróci wartość zmierzoną dla czasu CPU, czas zegarowy natomiast zostanie skrócony dla operacji arytmetycznych mnożenia.