

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone
Laboratorium 1.

Cel:

- przeprowadzenie pomiaru czasu CPU i zegarowego wykonania operacji
- organizacja środowiska tworzenia oprogramowania w systemie Linux (make, cc itp.)

Zajęcia 1:

1. Utworzenie katalogu roboczego (np. PR_lab).
2. Utworzenie podkatalogu roboczego (np. lab_1).
3. Skopiowanie do katalogu roboczego pliku „pomiar_czasu.tgz”
4. Rozpakowanie plików (`tar xvzf pomiar_czasu.tgz`): pliku Makefile sterującego programem „make”, kodu źródłowego procedur pomiaru czasu: `pomiar_czasu.c` oraz odpowiadającego pliku nagłówkowego: `pomiar_czasu.h`.
5. Pobranie programu `moj_program.c`, będącego prostym szablonem dla kodów źródłowych w C, zawierającego:
 - a) pętlę, w której realizowana jest operacja arytmetyczna
 - b) pętlę, w której realizowana jest operacja wejścia/wyjścia
6. Umożliwienie dokonania pomiaru czasu realizacji pętli poprzez:
 - a) włączenie pliku nagłówkowego `pomiar_czasu.h` z deklaracjami procedur pomiaru czasu
 - b) wywołanie przed wykonaniem pętli:
 - `inicjuj_czas()` (dla pętli pierwszej) oraz `t1=czas_zegara(); t2=czas_CPU()` (dla pętli drugiej)
 - c) wywołanie po wykonaniu pętli:
 - `drukuj_czas()` (dla pętli pierwszej) oraz `t1=czas_zegara()-t1; t2=czas_CPU()-t2` (dla pętli drugiej)
 - d) w efekcie jedna z pętli ma być mierzona za pomocą jednej wersji, a druga za pomocą drugiej
7. Modyfikacja pliku Makefile pozwalająca na skompilowanie napisanego programu z wykorzystaniem procedur pomiaru czasu
 - a) dodanie zależności dla końcowego pliku wykonywalnego i pliku pośredniego napisanego programu (plik pośredni ma być zależny także od pliku nagłówkowego „`pomiar_czasu.h`”, a plik wynikowy od pliku pośredniego „`pomiar_czasu.o`”)
 - b) dodanie poleceń umożliwiających utworzenie pliku wykonywalnego i pliku pośredniego dla napisanego programu
8. Kompilacja poleceniem „make”
9. Uruchomienie programu i dokonanie pomiaru czasu wykonania pętli za pomocą **obu** interfejsów.
10. Sformatowanie wydruku czasu realizacji drugiej pętli tak jak pierwszej (z wykorzystaniem funkcji `printf`).
11. Przeprowadzenie pomiaru czasu realizacji dla wersji do debugowania oraz wersji zoptymalizowanej.
12. Zgłoszenie wykonania pierwszej części zadania.
13. Analiza wyników: skąd biorą się różnice w czasach mierzonych dla obu pętli?

Dalsze kroki dla podniesienia oceny (w przypadku nie wykonania zadania w trakcie laboratorium należy je zrealizować przed kolejnymi zajęciami – jest to wymagane dla następnego tematu):

1. Przeniesienie plików związanych z pomiarem czasu do osobnego katalogu (o nazwie np. „pomiar_czasu”) na tym samym poziomie co `lab_1` (`lab_1` i `pomiar_czasu` powinny być podkatalogami katalogu roboczego `PR_lab`). W katalogu `lab_1` nie powinien pozostać żaden plik `pomiar_czasu.*`
2. Utworzenie w katalogu „pomiar_czasu” poleceniem „`ar -rs libpomiar_czasu.a pomiar_czasu.o`” statycznej biblioteki z procedurami pomiaru czasu (biblioteka i plik nagłówkowy pozostają w katalogu „pomiar_czasu”)
3. Modyfikacja pliku Makefile, tak aby umożliwiał korzystanie z biblioteki – dodanie opcji `-I` wskazującej na lokalizację plików nagłówkowych i `-L` wskazującej na lokalizację bibliotek oraz wykomentowanie fragmentów związanych z kompilacją `pomiar_czasu.c` i linkowaniem `pomiar_czasu.o` (należy też w pliku Makefile i kodzie źródłowym zmodyfikować lokalizację pliku nagłówkowego `pomiar_czasu.h`)
4. Wyczyszczenie poprzednich plików pośrednich poleceniem „`make clean`”
5. Ponowna kompilacja i wykonanie.

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie co najmniej kroków 1-12
2. Oddanie napisanego odręcznie sprawozdania z opisem zadania (cel, zrealizowane kroki, wnioski), kodem źródłowym procedury w C i zawartością pliku Makefile (obie wersje pliku Makefile – dla kompilacji z wykorzystaniem pliku źródłowego „`pomiar_czasu.c`” i dla korzystania z biblioteki „`libpomiar_czasu.a`”). We wnioskach zwrócić uwagę na różnice między czasem wykonania operacji arytmetycznych a czasem wykonania operacji we/wy oraz czasem CPU wykonywanych pętli i czasem zegara (zewnętrznym), a także czasami wersji do debugowania oraz wersji zoptymalizowanej.