Przemysław Szymoniak

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 2.

Clone – służy do tworzenia procesów poprzez klonowanie, czyli współdzielenie, przez ojca i potomka pewnych zasobów takich jak pamięci, tablicy deskryptorów, tablicy obsługi sygnałów. Jest wykorzystywany do implementacji wątków. Podczas tworzenia przesyłamy mu adres funkcji, która ma być przez niego wykonywana.

Fork – działa na zasadzie kopiowania stron pamięci, tworzy proces potomny różniący się jedynie PID oraz PPID dodatkowo użycie zasobów jest ustawione na 0.

Execv – służy do zastąpienia w pamięci obrazu aktualnego procesów przez obraz nowego procesów wskazanego jako plik przekazany do funkcji argumentem.

Celem laboratorium było przeprowadzenie pomiarów czasu CPU, zegarowego wykonywania operacji przy użyciu interfejsu procedur pomiaru czasu oraz nabycie umiejętności posługiwania się programami wykorzystującymi tworzenie wątków i procesów.

W ramach zajęć zrealizowałem następujące kroki:

- rozpakowanie archiwum „**pomiar_czasu.tgz**” oraz „**libpomiar_czasu.tgz”** oraz wykonanie polecenia **make**
- uzupełnienie plików źródłowych o procedury pomiaru czasu:

„fork.c”

```
#include "pomiar_czasu.h"
main(){
        ...

    int zmienna_globalna=0;
    inicjuj_czas();
    for(i=0;i<1000;i++){
        pid = fork();
        if(pid==0){

        ...

        }else {
            wait(NULL);
        }
    }
    drukuj_czas();
}
```

„clone.c”

```
#include "pomiar_czasu.h"
main(){
        ...

    inicjuj_czas();
    for(i=0;i<1000;i++){
        pid = clone( &funkcja_watku, (void *)
        stos+ROZMIAR_STOSU,
        CLONE_FS | CLONE_FILES |
        CLONE_SIGHAND | CLONE_VM, 0 );

        waitpid(pid, NULL, __WCLONE);
    }
    drukuj_czas();
    free( stos );
}
```

- kompilacja, uruchomienie programu oraz pomiar czasu tworzenia 1000 wątków oraz 1000 procesów

Pomiar czasu tworzenia 1000 procesów				
Lp.	**OPT = -g -DDEBUG**		**OPT = -O3**	
	Czas CPU	**Czas zegarowy**	**Czas CPU**	**Czas zegarowy**
1.	0.0	0.098405	0.003490	0.081794
2.	0.004525	0.126296	0.000951	0.073045
3.	0.001977	0.212009	0.0	0.076639
4.	0.001096	0.108878	0.001023	0.073714
5.	0.0	0.066611	0.002289	0.068215

Pomiar czasu tworzenia 1000 wątków				
Lp.	**OPT = -g -DDEBUG**		**OPT = -O3**	
	Czas CPU	**Czas zegarowy**	**Czas CPU**	**Czas zegarowy**
1.	0.001273	0.028788	0.001284	0.032568
2.	0.0	0.026536	0.001684	0.026196
3.	0.0	0.022901	0.000953	0.015653
4.	0.0	0.012485	0.0	0.032655
5.	0.000532	0.013958	0.000880	0.027466

- modyfikacja kodów programów przy wykorzystaniu funkcji **exec** wywołującej nowo stworzony program:

W pliku **„clone.c”** i **„fork.c”** od komentowanie linii:

```
int wynik;
wynik=execv("./program",NULL);
if(wynik==-1)
   printf("Proces potomny nie wykonal programu\n");
```

Stworzenie pliku **„program.c”:**

```
#include ...
main(){
  printf("Przemyslaw Szymoniak, pid: %d\n", getpid());
}
```

Oraz jego kompilacja do postaci **„./program”**

- modyfikacja programu, tak aby tworzyć po sobie dwa wątki równoległe, które zwiększają w pętli wartości dwóch zmiennych jednej globalnej, drugiej lokalnej :

```
int funkcja_watku( void* argument )
{
  for( int i=0;i<100;i++){
    zmienna_globalna++;
    argument++;
  }
  printf("zmienna globalna: %d, zmienna lokalna: %d
          \n", zmienna_globalna, argument);
  return 0;
}

main(){
  void *stos;
  void *stos2;
  int zmienna=0;
  pid_t pid;
  pid_t pid2;
  int i;
  stos = malloc( ROZMIAR_STOSU );
  stos2 = malloc( ROZMIAR_STOSU );

  if (stos == 0 || stos2 == 0) {
    printf("Proces nadrzędny - blad alokacji stosu\n");
    exit( 1 );   }
  inicjuj_czas();

  pid = clone( &funkcja_watku, (void * )
  stos+ROZMIAR_STOSU, CLONE_FS |
  CLONE_FILES | CLONE_SIGHAND | CLONE_VM,
  &zmienna);

  pid2 = clone( &funkcja_watku, (void *)
  stos2+ROZMIAR_STOSU, CLONE_FS |
  CLONE_FILES | CLONE_SIGHAND | CLONE_VM,
  &zmienna );

  waitpid(pid, NULL, __WCLONE);
  waitpid(pid2, NULL, __WCLONE);

  printf("koniec watku1 i zm. glob: %d, lok:%d\n",
          zmienna_globalna, zmienna);
  printf("koniec watku1 2 i zm. glob: %d, lok:%d\n",
          zmienna_globalna, zmienna);
  drukuj_czas();

  free( stos );
  free( stos2 );
}
```

Wnioski:

- przy wykorzystaniu **OPT = -g –DDEBUG** kompilator usuwał operacje nie wywołujące żadnych efektów, dlatego czas CPU dla procesów dwa razy wynosi 0.0 oraz dla wątków trzy razy wynosi 0.0
- przy wykorzystaniu **OPT = -O3** kompilator usunął tylko po jeden raz operacje nie wywołujące żadnych widocznych efektów dla czasu CPU w odniesieniu do wątków oraz procesów
- średni czas tworzenia 1000 procesów wynosi (ignorując pomiary zerowe):
  - ✓ **OPT = -g –DDEBUG:**
    - **dla CPU: 0.0025327**
    - **dla czasu zegarowego: 0.1224398**
  - ✓ średni czas tworzenia 1 procesu:
    - **dla CPU: 0.0000025**
    - **dla czasu zegarowego: 0.0001224**
  - ✓ stosunek czasu CPU/zegarowego dla 1 proc.:
    - **ok. 0.02 ≈ 2%**
  - ✓ **OPT = -O3:**
    - **dla CPU: 0.004078**
    - **dla czasu zegarowego: 0.0746814**
  - ✓ średni czas tworzenia 1 procesu:
    - **dla CPU: 0.0000040**
    - **dla czasu zegarowego: 0.0000747**
  - ✓ stosunek czasu CPU/zegarowego dla 1 proc.:
    - **ok. 0.05 ≈ 5%**
- średni czas tworzenia 1000 wątków wynosi (ignorując pomiary zerowe):
  - ✓ **OPT = -g –DDEBUG:**
    - **dla CPU: 0.0009025**
    - **dla czasu zegarowego: 0.0209336**
  - ✓ średni czas tworzenia 1 procesu:
    - **dla CPU: 0.0000009**
    - **dla czasu zegarowego: 0.0000209**
  - ✓ stosunek czasu CPU/zegarowego dla 1 wątk.:
    - **ok. 0.04 ≈ 4%**
  - ✓ **OPT = -O3:**
    - **dla CPU: 0.001200**
    - **dla czasu zegarowego: 0.0269076**
  - ✓ średni czas tworzenia 1 procesu:
    - **dla CPU: 0.0000012**
    - **dla czasu zegarowego: 0.0000269**
  - ✓ stosunek czasu CPU/zegarowego dla 1 wątk.:
    - **ok. 0.04 ≈ 4%**

- liczby obliczane w pętli dla zmiennej lokalnej oraz globalnej w trakcie trwania programu wyglądały identycznie dla poszczególnie wypisywanych linii
- aby polecenie **„clone”** działało jak **„fork”** należy wykorzystać flagi SIGCHLD|COPYVM