**Wojskowa Akademia Techniczna**

**Wydział Elektroniki**

**Instytut Telekomunikacji**

**Studia II°**

**Projektowanie aplikacji sieciowych**

Materiały pomocnicze do zajęć

Cz.2.

**dr inż. Jarosław Krygier**

Warszawa 2017

Spis treści

[1 Wątki 3](#_Toc495917770)

[2 Mutexy 7](#_Toc495917771)

# Wątki

/\*

============================================================================

Name : PAS\_6\_watki.c

Author : J. Krygier

Version :

Copyright :

Description : Watki

============================================================================

\*/

//

//Funkcje

/\*

pthread\_create Tworzenie watku

pthread\_exit Zakonczenie wątku biezącego

pthread\_join Czekanie na zakończenie watku

pthread\_self Pobranie identyfikatora biezacego watku

pthread\_cancel Kasowanie innego watku

.....................................................

pthread\_attr\_init Inicjacja atrybutów wątku

\*/

//UWAGA: dodac biblioteke do linkera: -l pthread (wlasciwosci-settings-linker libraries) lub w czasie kompilacji

//cc -o"PAS\_6\_watki" ./src/PAS\_6\_watki.o -l pthread

// wyswietlenie w konsoli watkow: ps -eL | grep PAS

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h> //fukcje watkow

#include <unistd.h> //sleep

// deklaracja funkcji obslugi watku

// przygotuj je w pierwszej kolejnosci

void\* fukcja\_obslugi\_watku1 ();

void\* fukcja\_obslugi\_watku2 ();

int main(void) {

puts("Watki: threads");

int i;

int kod\_powrotu; //zmienna do przechowywania wartosci fukcji pthread\_create ()

pthread\_t thread1, thread2; // id watkow: typ zmiennej zdefiniowany w <pthread.h>

int argument\_przekazany\_do\_watku1 = 10;

int argument\_przekazany\_do\_watku2 = 20;

int wynik; // wynik zwrocony przez watek

// utworz watek

/\* int pthread\_create(pthread\_t \*id, const pthread\_attr\_t \*attr, void\* (\*fun)(void\*), void\* arg)

\* id - identyfikator wątku;

\* attr - wskaźnik na atrybuty wątku, można podać NULL - zostaną użyte domyślne wartości;

\* fun - funkcja wykonywana w wątku; przyjmuje argument typu void\* i zwraca wartość tego samego typu;

\* arg - przekazywany do funkcji.

\*/

printf ("proces: %d, ID watku w0: %lu\n", getpid (), pthread\_self());

#if 0

pthread\_attr\_t attr; // atrybuty obslugi watku

int detachstate; // do sprawdzenia statusu zwolnienia pamiaci dla watka

//inicjacja atrybutow watku - gdy ich nie bedzie mozna przyjac standardowe (przekazac NULL do 'pthread\_create')

pthread\_attr\_init(&attr);

pthread\_attr\_getdetachstate(&attr, &detachstate); // sprawdzenia standardowego statusu zwolnienia pamiaci dla watka

if (detachstate == PTHREAD\_CREATE\_JOINABLE)

printf ("Standardowy argument: PTHREAD\_CREATE\_JOINABLE\n"); // pamiec watkow sie zwolni po osiagnieciu funkcji 'pthread\_join'

else if (detachstate == PTHREAD\_CREATE\_JOINABLE)

printf ("Standardowy argument: PTHREAD\_CREATE\_DETACHED\n"); // pamiec watkow sie zwolni zaraz po zakonczeniu watka 'pthread\_exit'

else

printf ("w zasadzie innego satusu odlaczenia nie ma i to nie zostanie wypisane\n");

//moge zmianic status zwalniania pamieci poprzez zmiane atrybutu

pthread\_attr\_setdetachstate (&attr, PTHREAD\_CREATE\_DETACHED);

// i przekazanie attr do 'pthread\_create'

// kod\_powrotu = pthread\_create( &thread1, &attr, &fukcja\_obslugi\_watku1, &argument\_przekazany\_do\_watku1);

// po czym zwolnienie pamieci atrybutu:

pthread\_attr\_destroy(&attr);

#endif

kod\_powrotu = pthread\_create( &thread1, NULL, &fukcja\_obslugi\_watku1, &argument\_przekazany\_do\_watku1);

if(kod\_powrotu){

printf("Error - pthread\_create() zwraca kod: %d\n",kod\_powrotu);

exit(EXIT\_FAILURE);

} else {

printf("watek1: %lu pthread\_create() zwraca: %d\n", (unsigned long int) thread1, kod\_powrotu);

}

kod\_powrotu = pthread\_create( &thread2, NULL, &fukcja\_obslugi\_watku2, &argument\_przekazany\_do\_watku2);

if(kod\_powrotu){

printf("Error - pthread\_create() zwraca kod: %d\n",kod\_powrotu); // kod powrotu inny niz 0 jesli blad

exit(EXIT\_FAILURE);

} else {

printf("watek2: %lu pthread\_create() zwraca: %d\n", (unsigned long int) thread2, kod\_powrotu); // kod powrotu 0 jesli sukces

}

#if 0

//skasowanie innego watku

//wykonac na koniec

//unikac kasowania innych watkow przed ich zakonczeniem

pthread\_cancel(thread1);

pthread\_cancel(thread2);

#endif

for (i=0; i< 30; i++) {

printf ("w0\n");

sleep (1);

}

//poczekamy na zakonczenie watkow w1 i w2 przed zakonczeniem funkcji main (watku 0)

pthread\_join( thread1, NULL);

// watek w1 nie zwrocil nic

pthread\_join( thread2, (void\*)&wynik);

// watek w2 zwrocil wynik

printf ("Po skonczeniu watek 2 zwrocil: %d \n", wynik);

for (i=0; i< 5; i++) {

printf ("w0\n");

sleep (1);

}

printf ("Koncze program glowny (watek 0) \n");

return EXIT\_SUCCESS;

}

void\* fukcja\_obslugi\_watku1 (void \*arg) {

int \*ptr;

ptr = (int \*) arg;

int i;

int liczba\_kropek = 0;

pthread\_t id\_watku;

liczba\_kropek = (\*ptr);

printf ("Wykonuje watek 1 (liczba przekazana:%d)\n", liczba\_kropek);

// pobierz id tego watku

id\_watku = pthread\_self();

printf ("proces: %d, ID watku w1: %lu\n", getpid (), id\_watku); // id\_watku jest (unsigned long int), dlatego 'lu'

sleep (1);

for (i=0; i< liczba\_kropek; i++) {

printf ("w1\n");

sleep (1);

}

printf ("Koncze watek 1\n");

return NULL;

}

void\* fukcja\_obslugi\_watku2 (void \*arg) {

int \*ptr;

ptr = (int \*) arg;

int i;

int liczba\_kropek = 0;

int wynik;

pthread\_t id\_watku;

liczba\_kropek = (\*ptr);

printf ("Wykonuje watek 2 (liczba przekazana:%d)\n", liczba\_kropek);

// pobierz id tego watku

id\_watku = pthread\_self();

printf ("proces: %d, ID watku w2: %lu \n", getpid (), id\_watku);

sleep (1);

#if 0

//wczesniejsze zakoczenie watku

printf ("Przerwalem watek 2 'pthread\_exit()' \n");

pthread\_exit(NULL);

#endif

//lub

#if 0

//wczesniejsze zakoczenie watku

printf ("Przerwalem watek 2 (RETURN) \n");

return NULL;

#endif

for (i=0; i< liczba\_kropek; i++) {

printf ("w2\n");

sleep (1);

}

wynik = 100;

printf ("Koncze watek 1\n");

#if 0

//zwrocenie wyniki przez watek

return ((void\*) wynik);

#endif

// lub

#if 1

//zwrocenie wyniki przez watek

pthread\_exit((void\*) wynik);

#endif

}

# Mutexy

/\*============================================================================

Name : PAS\_6A\_mutex.c

Author : J. Krygier

Version :

Copyright :

Description : Watki Mutexy

============================================================================

\*/

#include <pthread.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#define WLACZ\_MUTEX 0

int zmiennaglobalna=0;

#if WLACZ\_MUTEX

pthread\_mutex\_t mojmuteks=PTHREAD\_MUTEX\_INITIALIZER;

#endif

void \*Watek1(void \*arg) {

int i;

for ( i=0; i<20; i++ ) {

#if WLACZ\_MUTEX

pthread\_mutex\_lock(&mojmuteks);

#endif

printf("1(%d)\n", zmiennaglobalna);

fflush(stdout);

zmiennaglobalna=zmiennaglobalna+1;

sleep(1);

#if WLACZ\_MUTEX

pthread\_mutex\_unlock(&mojmuteks);

#endif

}

return NULL;

}

int main(void) {

pthread\_t w1;

int i;

if ( pthread\_create( &w1, NULL, Watek1, NULL) ) {

printf("Blad przy tworzeniu watku.");

abort();

}

for ( i=0; i<20; i++) {

#if WLACZ\_MUTEX

pthread\_mutex\_lock(&mojmuteks);

#endif

printf("0(%d)\n", zmiennaglobalna);

fflush(stdout);

zmiennaglobalna=zmiennaglobalna-1;

#if WLACZ\_MUTEX

pthread\_mutex\_unlock(&mojmuteks);

#endif

sleep(1);

}

if ( pthread\_join (w1, NULL ) ) {

printf("Blad przy tworzeniu watku.");

abort();

}

printf("\nMoja zmienna globalna wynosi %d\n",zmiennaglobalna);

exit(0);

}