**2 ЛАБОРАТОРНАЯ РОБОТА №2**

**ПЛАНУВАЛЬНИК ПОТОКІВ У ПРОСТОРІ КОРИСТУВАЧА**

**2.1 Мета роботи**

Ознайомитися з поняттям планувальник, з алгоритмом планування, розглянути основні види планувальників.

**2.2 Виконання роботи**

#ifndef CONTROLLER\_H

#define CONTROLLER\_H

#define NEXTTHREAD 1

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include "thread.h"

#include <QList>

void shedule();

void add\_thread(thread\_t \*ptr);

void add\_sleep(thread\_t \*ptr);

void chek\_speep();

void\* main\_func(void\*);

void init\_controller();

void start\_controller();

void next\_itr();

void timer\_handler(int, siginfo\_t\*, void\*);

void init\_timer();

void timer\_lock();

void timer\_unlock();

thread\_t \*get\_current();

void set\_current(thread\_t \*ptr);

void run\_thread(thread\_t \*ptr);

#endif // CONTROLLER\_H

#include "controller.h"

#include <QDebug>

QList<thread\_t \*> l\_threads;

QList<thread\_t \*> l\_sleep;

thread\_t \*curent;

thread\_t \*main\_thread;

void timer\_handler(int arg, siginfo\_t \*inf, void \*ptr)

{

shedule();

}

void next\_itr()

{

thread\_t \*ptr = get\_current();

if(ptr->state == RUN) {

ptr->state = READY;

l\_threads.append(ptr);

}

alarm(NEXTTHREAD);

thread\_t \*next;

if(l\_threads.size() == 0) {

next = main\_thread;

} else {

next = l\_threads.takeFirst();

}

run\_thread(next);

}

void init\_controller()

{

create\_thread(main\_func, NULL);

main\_thread = l\_threads.takeFirst();

main\_thread->state = MAIN;

curent = main\_thread;

init\_timer();

}

void init\_timer() {

struct sigaction sa;

memset(&sa, 0, sizeof(struct sigaction));

sa.sa\_sigaction = timer\_handler;

sa.sa\_flags = SA\_SIGINFO | SA\_NODEFER;

sigaction(SIGALRM, &sa, NULL);

}

void timer\_lock(){

sigset\_t set;

sigemptyset(&set);

sigaddset(&set, SIGALRM);

sigprocmask(SIG\_BLOCK, &set, NULL);

}

void timer\_unlock(){

sigset\_t set;

sigemptyset(&set);

sigaddset(&set, SIGALRM);

sigprocmask(SIG\_UNBLOCK, &set, NULL);

}

thread\_t \*get\_current()

{

return curent;

}

void \*main\_func(void \*)

{

qDebug() << "RUN CONTROLLER";

while (1) {}

return NULL;

}

void start\_controller()

{

qDebug() << "start\_controller";

alarm(NEXTTHREAD);

setcontext(main\_thread->context);

}

void set\_current(thread\_t \*ptr)

{

curent = ptr;

}

void add\_thread(thread\_t \*ptr)

{

l\_threads.append(ptr);

}

void shedule()

{

chek\_speep();

next\_itr();

}

void add\_sleep(thread\_t \*ptr)

{

l\_sleep.append(ptr);

}

void chek\_speep()

{

for (int i = 0; i < l\_sleep.size(); ++i) {

thread\_t \*ptr = l\_sleep.at(i);

if(ptr->wake\_time > time(NULL)) {

ptr->state = READY;

l\_sleep.removeAt(i);

l\_threads.append(ptr);

}

}

}

**2.3 Висновки**

У ході виконання даної лабораторної роботи було ознайомлено з поняттям планувальник, з алгоритмом планування, розглянуті основні види планувальників.