ЛАБОРАТОРНА РОБОТА 2

Створення статичних і динамічних бібліотек для POSIX сумісних (сертифікованих) операційних систем

Мета роботи полягає у оволодінні практичними навичками розробки ПО на системному рівні з використанням бібліотек мови С (С++) для POSIX OS, на прикладі ОС Linux та автоматизації розробки з використанням Makefile статичних і динамічних бібліотек.

Другою метою є ознайомлення з варіантами реалізації алгоритмів у процедурному і ООП парадигмах з використанням мови С (С++).

Перед початком розробки програм потрібно вивчити: основи синтаксису статичних і динамічних бібліотек; організацію і структуру файлової системи POSIX OS, обмеження на імена файлів; типи файлів, каталоги і посилання [1], синтаксис Makefile.

Завдання для лабораторної роботи 2

1. Розробити консольний додаток у парадигмі ООП, що виконує наступні функції:

- додавання “Робітника” до інформаційній системи або іншу інформаційну сутність за спрямованістю, що вказана у ЛР1 і за результатами ЛР1, далі просто “Робітника”;

- видалення “Робітника” зі штату, при цьому він має залишатися у інформаційній системи, як минулий;

- збільшення (зменшення) зарплатні “Робітника”, або іншу аналогічну інформаційну сутність за тематичною за спрямованістю, що вказана у ЛР1 відповідно до варіанту;

- зображення інформації про “Робітника”, або іншу аналогічну інформаційну сутність за тематичною за спрямованістю, що вказана у ЛР1 відповідно до варіанту.

Для розробки скористатися вихідним кодом та інформацією, що вказана у [5]. Реалізація вихідного коду показана у додатку 2.

2. Створити два репозіторія, користуючись інформацією [1]. У одному програма має бути розділена на основний коді і статичні бібліотеки. У іншому репозиторії розробити динамічні бібліотеки.

3. За бажанням можна реалізувати функціональність п.1 у чистому стилі ООП поза шаблоном [5] самостийно. Або реалізувати у процедурному варіанті.

Результатом роботи мають бути 2 репозиторія. У першому 5 файлів вихідного коду відповідно додатку 1 і Makefile для створення динамічних бібліотек.

У другому репозиторії 5 файлів вихідного коду відповідно додатку 1 і Makefile для створення статичних бібліотек.

Програма проведення експерименту лабораторної роботи

1. Створення вихідного коду. Користуючись одним з текстових редакторів, створити файл вихідного коду лабораторної роботи. Зберегти його у робочому каталозі, що містить програмну. Доробити вихідний код відповідно до поданого завдання.
2. У разі появи помилок внести зміни до вихідного коду.
3. Після успішного виконання п. 1, 2 запустити програму на виконання.
4. Після покрокового виконання програми зробити скриншоти екранів і створити звіт, додаючи до звіту скриншоти екранів.
5. Зробити висновки щодо особливостей виконання лабораторної роботи, наявності специфічних особливостей роботи програми. Відмінностей статичних і динамічних бібліотек.

Теоретичні відомості для ЛР 2

Приклад статичної бібліотеки

Створимо свою власну бібліотеку, що має функції: h_world () і g_world (), і виводять на екран "Hello World" і "Goodbye World" відповідно. Почнемо зі статичної бібліотеки.

Створимо файл world.h:

```
/ * World.h * /
void h_world (void);
void g_world (void);
```

Створимо файл h_world.c:

```
/ * H_world.c * /
#include <stdio.h>
#include "world.h"

void h_world (void)
{
      printf ( "Hello World \ n");
```

```
}
```

Тепер створимо файл g_world.c, що містить реалізацію функції g_world ():

```
/ * G_world.c * /
#include <stdio.h>
#include "world.h"
void g_world (void){
    printf ( "Goodbye World \ n");}
```

Можна було б з таким же успіхом вмістити обидві функції в одному файлі (hello.c, наприклад), проте для наочності ми рознесли код на два файли.

Тепер створимо файл main.c. Це клієнт, який буде користуватися послугами сервера:

```
/ * Main.c * /
#include "world.h"
int main (void){
    h_world ();
    g_world ();
}
```

Тепер напишемо сценарій для make. Для цього створюємо Makefile:

```
# Makefile for World project
binary: main.o libworld.a
    gcc -o binary main.o -L. -lworld
main.o: main.c
    gcc -c main.c
libworld.a: h_world.o g_world.o
    ar cr libworld.a h_world.o g_world.o
h_world.o: h_world.c
    gcc -c h_world.c
g_world.o: g_world.c
    gcc -c g_world.c
clean:
    rm -f * .o * .a binary
```

У прикладі нижче для наочності у репозиторії присутні дві статичні бібліотеки.

```
	binary:main.o test_lib.a
	 gcc -o binary main.o -L. -l:   test_lib.a
main.o:main.c
	 gcc -c main.c
test_lib.a:   f1.o   f2.o
	 ar cr test_lib.a f1.o f2.o
f1.o:   f2.c
	 gcc -c f1.c
f2.o:   f2.c
	 gcc -c f2.c
clean:
	 rm -f *.o *.a binary
```

Збираємо програму:

```
$ make
gcc -c main.c
gcc -c h_world.c
gcc -c g_world.c
ar cr libworld.a h_world.o g_world.o
gcc -o binary main.o -L. -lworld
$
```

Перевіряємо результат

```
$ ./binary
Hello World
Goodbye World
```

В наведеному прикладі з'явилися три нові речі: опції -l і -L компілятора, а також команда ar. Команда ar створює статичну бібліотеку (архів). У нашому випадку два об'єктних файлу об'єднуються в один файл libworld.a. У Linux практично всі бібліотеки мають префікс lib.

Опція -l, передана компілятору, обробляється і надсилається лінковщік для того, щоб той підключив до бінарники бібліотеку. Важливо знати лише те, що і бібліотека

libfoo.so і бібліотека libfoo.a підключаються до проекту опцією -lfoo. У нашому випадку libworld.a "урізані" до -lworld.

Опція -L вказує лінковщік, де йому шукати бібліотеку. У разі, якщо бібліотека розташовується в каталозі / lib або / usr / lib, то питання відпадає сам собою і опція -L не потрібна. У нашому випадку бібліотека знаходиться в репозиторії (в поточному каталозі). За замовчуванням лінковщік не проглядається поточний каталог в пошуку бібліотеки, тому опція -L необхідна.

Дінамічна бібліотека

Для того, щоб створити і використовувати динамічну (спільно використовувану) бібліотеку, досить потрібно переробити в нашому проекті Makefile.

```
# Makefile for World project

binary: main.o libworld.so
gcc -o binary main.o -L. -lworld -Wl, -rpath ,.
main.o: main.c
gcc -c main.c
libworld.so: h_world.o g_world.o
gcc -shared -o libworld.so h_world.o g_world.o
h_world.o: h_world.c
gcc -c -fPIC h_world.c
g_world.o: g_world.c
gcc -c -fPIC g_world.c
clean:
rm -f * .o  *.so binary
```

У прикладі нижче для наочності у репозиторії присутні дві динамічні бібліотеки.

```
binary:main.o  test_lib.so
gcc -o binary main.o -L. -l: test_lib.so -Wl,--rpath -Wl,.
main.o:main.c
gcc -c main.c
```

```
test_lib.so: f1.o f2.o
	gcc -shared -o  test_lib.so f1.o f2.o
f1.o: f1.c
	gcc -c -fPIC f1.c
f2.o: f2.c
	gcc -c -fPIC f2.c
clean:
	rm -f *.o *.a binary
```

Правило для збірки binary тепер містить опцію -Wl, -rpath. Вона означає: передати лінковщік (-Wl) опцію option з аргументами optargs. У нашому випадку ми передаємо лінковщік опцію -rpath з аргументом. Цо означає опція -rpath? Як вже говорилося, лінковщік шукає бібліотеки в певних місцях, зазвичай це каталоги / lib і / usr / lib, іноді / usr / local / lib. Опція -rpath просто додає до цього списку ще один каталог. У нашому випадку це поточний каталог. Без вказівки опції -rpath програма не запуститься через відсутність бібліотеки.

Питання до ЛР2

1. Базовий перелік команд оболонки для управління і роботи у POSIX сумісних (сертифікованих) ОС.
2. Права доступу до файлів у типових POSIX сумісних (сертифікованих) ОС. Способи відображення і описання. Способи зміни прав доступу.
3. Основні правила розробки Makefile для створення статичних і динамічних бібліотек.
4. Відміннсті статичних і динамчних бібліотек у POSIX сумісних (сертифікованих) ОС.
5. Основні команди для всіх етапів розробки програм у GCC компіляторі.
6. Описання роботи розробленої програми на алгоритмічному рівні.
7. Синтаксис класу у C++, створення екземпляру класу, рівні доступу, управління рівнями доступу, доступ до методів екземпляру класу.

Література

1. https://www.opennet.ru/docs/RUS/zlp/
2. Б.Керниган, Д.Ритчи, А.Фьюер. Язык программирования Си. Задачи по языку Си. М.: Финансы и статистика, 1985.
3. https://riptutorial.com/Download/cplusplus.pdf
4. https://ru.stackoverflow.com/questions/432480/Создание-makefile
5. Николас А. Солтер, Майкл Л. Клепер «C++ для профессионалов»
6. https://linuxguide.rozh2sch.org.ua/

7. Операційні системи. Методичні вказівки до виконання комп’ютерного практикуму. Для студентів навчальних напрямів 6.040204 «Прикладна математика», 6.170101 «Безпека інформаційних і комунікаційних систем» /Укладачі Грайворонський М.В., Ільїн М.І., Родіонов А.М. – Київ, НТУУ «КПІ», 2012 - 58 с.
8. Maurice J. Bach THE DESIGN OF THE UNX OPERATING SYSTEM

https://konspect.blogspot.com/2013/10/virtualbox-debian-nastrojka-obshhih.html
https://hyperhost.ua/info/ru/rabota-s-redaktorom-nano-bazovyie-klavishn
http://www.gdbtutorial.com/tutorial/how-install-gdb
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_gdb.php
http://www.gdbtutorial.com/tutorial/how-use-gdb-example

Тематична спрямованість для створення класів ЛР 1. Додаток 1

№1: Інформаційна система Бібліотеки.

Клас

Працівники (Код співробітника, ПІБ, Вік, Пол, Адреса, Паспортні дані, Код посади) [10 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№2: Інформаційна система Радіостанції.

Клас

Посади (Код посади, Найменування посади, Оклад, Обов'язки, Вимоги) [5 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№3: Інформаційна система таксопарку

Клас

Автомобілі (Код автомобіля, Код марки, Реєстраційний номер, Номер кузова, Номер двигуна, Рік випуску, Пробіг, Код шофѐра співробітника, Дата останнього ТО, Код-механіка співробітника, Спеціальні позначки) [10 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і

на стеку.

№4: Інформаційна система Туристичного агентства

Клас

Готелі (Код готелю, Найменування, Країна, Місто, Адреса, Кількість звѐзд, Контактна особа) [10 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№5: Інформаційна система Страхової компанії

Клас

Поліси (Номер поліса, Дата початку, Дата закінчення, -імость Сто, Сума виплати, Код виду поліса, Відмітка про виплату, Відмітка про закінчення, Код клієнта, Код співробітника) [10 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№6: Інформаційна система Шлюбного агентства

Клас

Клієнти (Код клієнта, ПІБ, Пол, Дата народження, Вік, Зростання, вага, Кількість дітей, Сімейний стан, Шкідливі при-вичкі, Хобі, Опис, Код знака, Код відносини, Код націо-нальності, Адреса, Телефон, Паспортні дані , Інформація про партнѐре) [10 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№7: Інформаційна система Сервіс центру

Клас

Види несправностей (Код виду, Код моделі, Опис, Симптоми, Методи ремонту, запчастини Код 1, Код запчастини 2, 3 Код запчастини, Ціна роботи) [5 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

№8: Інформаційна система Школи

Клас

Класи (Код класу, Код співробітника-класного керівника, Код виду, Кількість учнів, Буква, Рік навчання, Рік створення) [5 записів].

Передбачити функціональність додавання інформації, вивід інформації на консоль.

Передбачити функціональність поведінки класу, в залежності від значення певних членів класу.

Під час тестування на етапі виконання створити екземпляри класу на купі пам'яті і на стеку.

Додаток 2. Інформаційна система. Містить 5 файлів.

// Database.h

```
#include <iostream>
#include "Employee.h"

namespace Records {

  const int kMaxEmployees = 100;
  const int kFirstEmployeeNumber = 1000;
    class Database
      {
      public:
          Database();
          ~Database();

          Employee& addEmployee(std::string inFirstName, std::string inLastName);
          Employee& getEmployee(int inEmployeeNumber);
          Employee& getEmployee(std::string inFirstName, std::string inLastName);
          void       displayAll();
          void       displayCurrent();
          void       displayFormer();
      protected:
          Employee    mEmployees[kMaxEmployees];
          int         mNextSlot;
          int         mNextEmployeeNumber;
      };
}
```

// Database.cpp

```
#include <iostream>
#include <stdexcept>
#include <string>

#include "Database.h"

using namespace std;

namespace Records {

  Database::Database()
  {
    mNextSlot = 0;
    mNextEmployeeNumber = kFirstEmployeeNumber;
  }
  Database::~Database()
  {
  }
  Employee& Database::addEmployee(string inFirstName, string inLastName)
  {
    if (mNextSlot >= kMaxEmployees) {
      cerr << "There is no more room to add the new employee!" << endl;
      throw exception();
```

```
  }

  Employee& theEmployee = mEmployees[mNextSlot++];
  theEmployee.setFirstName(inFirstName);
  theEmployee.setLastName(inLastName);
  theEmployee.setEmployeeNumber(mNextEmployeeNumber++);
  theEmployee.hire();

  return theEmployee;
}
Employee& Database::getEmployee(int inEmployeeNumber)
{
  for (int i = 0; i < mNextSlot; i++) {
    if (mEmployees[i].getEmployeeNumber() == inEmployeeNumber) {
        return mEmployees[i];
    }
  }

  cerr << "No employee with employee number " << inEmployeeNumber << endl;
  throw exception();
}

Employee& Database::getEmployee(string inFirstName, string inLastName)
{
  for (int i = 0; i < mNextSlot; i++) {
    if (mEmployees[i].getFirstName() == inFirstName &&
          mEmployees[i].getLastName() == inLastName) {
        return mEmployees[i];
    }
  }

  cerr << "No match with name " << inFirstName << " " << inLastName << endl;
  throw exception();
}
void Database::displayAll()
{
  for (int i = 0; i < mNextSlot; i++) {
    mEmployees[i].display();
  }
}

void Database::displayCurrent()
{
  for (int i = 0; i < mNextSlot; i++) {
    if (mEmployees[i].getIsHired()) {
        mEmployees[i].display();
    }
  }
}

void Database::displayFormer()
{
```

```
    for (int i = 0; i < mNextSlot; i++) {
      if (!mEmployees[i].getIsHired()) {
          mEmployees[i].display();
      }
    }
  }
}
```

// Employee.h

```
#include <iostream>

namespace Records {
  const int kDefaultStartingSalary = 30000;
  class Employee
    {
    public:

      Employee();

      void     promote(int inRaiseAmount = 1000);
      void     demote(int inDemeritAmount = 1000);
      void     hire();     // hires or re-hires the employee
      void     fire();     // dismisses the employee
      void     display();  // outputs employee info to the console

      // Accessors and setters
      void          setFirstName(std::string inFirstName);
      std::string   getFirstName();
      void          setLastName(std::string inLastName);
      std::string   getLastName();
      void          setEmployeeNumber(int inEmployeeNumber);
      int           getEmployeeNumber();
      void          setSalary(int inNewSalary);
      int           getSalary();
      bool          getIsHired();
    private:
      std::string   mFirstName;
      std::string   mLastName;
      int           mEmployeeNumber;
      int           mSalary;
      bool          fHired;
    };
}
```

// Employee.cpp

```
#include <iostream>
#include <string>

#include "Employee.h"
```

```
using namespace std;

namespace Records {

 Employee::Employee()
 {
  mFirstName = "";
  mLastName = "";
  mEmployeeNumber = -1;
  mSalary = kDefaultStartingSalary;
  fHired = false;
 }
 void Employee::promote(int inRaiseAmount)
 {
  setSalary(getSalary() + inRaiseAmount);
 }

 void Employee::demote(int inDemeritAmount)
 {
  setSalary(getSalary() - inDemeritAmount);
 }
 void Employee::hire()
 {
  fHired = true;
 }

 void Employee::fire()
 {
  fHired = false;
 }
 void Employee::display()
 {
  cout << "Employee: " << getLastName() << ", " << getFirstName() << endl;
  cout << "-------------------------" << endl;
  cout << (fHired ? "Current Employee" : "Former Employee") << endl;
  cout << "Employee Number: " << getEmployeeNumber() << endl;
  cout << "Salary: $" << getSalary() << endl;
  cout << endl;
 }
 // Accessors and setters

 void Employee::setFirstName(string inFirstName)
 {
  mFirstName = inFirstName;
 }

 string Employee::getFirstName()
 {
  return mFirstName;
 }

 void Employee::setLastName(string inLastName)
```

```
  {
    mLastName = inLastName;
  }

  string Employee::getLastName()
  {
    return mLastName;
  }

  void Employee::setEmployeeNumber(int inEmployeeNumber)
  {
    mEmployeeNumber = inEmployeeNumber;
  }

  int Employee::getEmployeeNumber()
  {
    return mEmployeeNumber;
  }

  void Employee::setSalary(int inSalary)
  {
    mSalary = inSalary;
  }

  int Employee::getSalary()
  {
    return mSalary;
  }

  bool Employee::getIsHired()
  {
    return fHired;
  }

}
```

// UserInterface.cpp

```
#include <iostream>
#include <stdexcept>
#include <string>

#include "Database.h"

using namespace std;
using namespace Records;

int displayMenu();
void doHire(Database& inDB);
void doFire(Database& inDB);
void doPromote(Database& inDB);
void doDemote(Database& inDB);
```

```
int main(int argc, char** argv)
{
  Database employeeDB;
  bool done = false;

  while (!done) {
    int selection = displayMenu();

    switch (selection) {
    case 1:
      doHire(employeeDB);
      break;
    case 2:
      doFire(employeeDB);
      break;
    case 3:
      doPromote(employeeDB);
      break;
    case 4:
      employeeDB.displayAll();
      break;
    case 5:
      employeeDB.displayCurrent();
      break;
    case 6:
      employeeDB.displayFormer();
      break;
    case 0:
      done = true;
      break;
    default:
      cerr << "Unknown command." << endl;
    }
  }
}

int displayMenu()
{
  int selection;

  cout << endl;
  cout << "Employee Database" << endl;
  cout << "-----------------" << endl;
  cout << "1) Hire a new employee" << endl;
  cout << "2) Fire an employee" << endl;
  cout << "3) Promote an employee" << endl;
  cout << "4) List all employees" << endl;
  cout << "5) List all current employees" << endl;
  cout << "6) List all previous employees" << endl;
  cout << "0) Quit" << endl;
  cout << endl;
```

```
  cout << "---> ";

  cin >> selection;

  return selection;
}

void doHire(Database& inDB)
{
  string firstName;
  string lastName;

  cout << "First name? ";
  cin >> firstName;
  cout << "Last name? ";
  cin >> lastName;

  try {
    inDB.addEmployee(firstName, lastName);
  } catch (std::exception ex) {
    cerr << "Unable to add new employee!" << endl;
  }
}

void doFire(Database& inDB)
{
  int employeeNumber;

  cout << "Employee number? ";
  cin >> employeeNumber;

  try {
    Employee& emp = inDB.getEmployee(employeeNumber);
    emp.fire();
    cout << "Employee " << employeeNumber << " has been terminated." << endl;
  } catch (std::exception ex) {
    cerr << "Unable to terminate employee!" << endl;
  }
}

void doPromote(Database& inDB)
{
  int employeeNumber;
  int raiseAmount;

  cout << "Employee number? ";
  cin >> employeeNumber;

  cout << "How much of a raise? ";
  cin >> raiseAmount;

  try {
```

```
    Employee& emp = inDB.getEmployee(employeeNumber);
    emp.promote(raiseAmount);
  } catch (...) {
    cerr << "Unable to promote employee!" << endl;
  }
}
```

Додаток до ЛР 6

Вихідний код модуля ядра. Частина перша.

```
#include <linux/kernel.h>
#include <linux/module.h>

MODULE_AUTHOR("Present <https://github.com/danya-psch>");  // Подарунок
MODULE_DESCRIPTION("Lab1 test kernel module"); // Опис фунціоналу
MODULE_LICENSE("MIT"); // ліцензія

/*
 (hello_init)
 (hello_exit).
 module_init i
module_exit використовують специальні макросы ядра
*/

static int __init hw_init(void)
{
        printk(KERN_ERR "Input , Kernel!\n");
        return 0;
}

static void __exit hw_exit(void)
{
        printk(KERN_ERR "It out from, Kernel!\n");
}
module_init(hw_init);
module_exit(hw_exit);
```

Зміст Makefile. Частина перша.

```
ifneq ($(KERNELRELEASE),)
# kbuild part of makefile
obj-m := hw_module.o
else
# normal makefile
KDIR ?= /lib/modules/`uname -r`/build

default:
        $(MAKE) -C $(KDIR) M=$$PWD
clean:
        $(MAKE) -C $(KDIR) M=$$PWD clean
endif
```

Вихідний код модуля ядра. Частина друга.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/random.h>
#include <linux/slab.h>
#include <linux/types.h>

MODULE_AUTHOR("Danya Peschanskyi<https://github.com/danya-psch>");
MODULE_DESCRIPTION("Lab 2, using struct list_head");
MODULE_LICENSE("MIT");

#define LIST_LEN 10
#define MSG_PREF "DANYA_MSG: "

#define print_msg(msg, ...) printk(KERN_ERR MSG_PREF msg, ##__VA_ARGS__);

typedef struct {
        struct list_head lnode;
        uint32_t val;
} int_node_t;

// макрос для освобождения памяти выделеной под элементы списка,
// do {}while(0) требуеться, чтоб обернуть код макроса в блок, простого {} недостаточно
#define ilfree(list_head) \
   do {                                                \
      int_node_t *__ptr, *__tmp;                       \
      list_for_each_entry_safe(__ptr, __tmp, (list_head), lnode) {     \
         kfree(__ptr);                                 \
      }                                                \
   } while(0)

// макрос для вывода значений списка в консоль
#define ilprint(list_head) \
   do {                                                \
      int_node_t *__ptr;                               \
      print_msg("List: {");                            \
      list_for_each_entry(__ptr, (list_head), lnode) {              \
         printk(KERN_ERR "\t%i ", __ptr->val);                      \
      }                                                \
      printk(KERN_ERR "}\n");                          \
   } while(0)

// инициализация головы списка, всю реализацию списка можно найти по ссылке
https://elixir.bootlin.com/linux/latest/source/include/linux/list.h#L714
static struct list_head int_list = LIST_HEAD_INIT(int_list);

// функция которая побитово сдвигает значение каждого элемента списка на 1 влево
static void task(void) {
        int_node_t *ptr;
        list_for_each_entry(ptr, &int_list, lnode) {
```

```
                ptr->val <<= 1;
        }
}

static int __init list_module_init(void)
{
        int i;
        int ret = 0;
        print_msg("List allocation start...\n");
        // выделяем динамически память для элементов списка в цыкле
        for (i = 0; i < LIST_LEN; ++i) {
                int_node_t *ptr = (int_node_t *)kmalloc(sizeof(*ptr), GFP_KERNEL);
                if (!ptr) {
                        print_msg("Can't alloc memory\n");
                        ret = -ENOMEM;
                        goto alloc_err;
                }
                // функция которая дает нам рандомное значение для указанного количества
байт
                get_random_bytes(&ptr->val, sizeof(ptr->val));
                list_add_tail(&ptr->lnode, &int_list);
        }
        print_msg("List allocation finish\n");
        ilprint(&int_list);

        task();

        ilprint(&int_list);

        return 0;
alloc_err:
        ilfree(&int_list);
        return ret;
}

static void __exit list_module_exit(void)
{
        ilfree(&int_list);
        print_msg("Hasta la vista, Kernel!\n");
}
module_init(list_module_init);
module_exit(list_module_exit);
```

Makefile модуля ядра. Частина друга.

```
ifneq ($(KERNELRELEASE),)
# kbuild part of makefile
obj-m := list_module.o
else
# normal makefile
KDIR ?= /lib/modules/`uname -r`/build
```

```
default:
	$(MAKE) -C $(KDIR) M=$$PWD
clean:
	$(MAKE) -C $(KDIR) M=$$PWD clean
endif
```

Додаток до ЛР 7

Вихідний код модуля ядра.

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kdev_t.h>
#include <linux/fs.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/time.h>
#include <linux/uaccess.h>

MODULE_AUTHOR("Danya Peschanskyi<https://github.com/danya-psch>");
MODULE_DESCRIPTION("Simple char device module");
MODULE_LICENSE("GPL");

#define BUF_LEN 256

#define MSG_PREF "DANYA_MSG: "
#define print_msg(msg, ...) printk(KERN_ERR MSG_PREF msg, ##__VA_ARGS__);

// структуры ядра необходимые для создания символьного драйвера
dev_t devt = 0;
static struct class *dev_class = NULL;
static struct cdev my_cdev;
static struct device *dev = NULL;

static int      my_open(struct inode *inode, struct file *file);
static int      my_release(struct inode *inode, struct file *file);
static ssize_t  my_read(struct file *filp, char __user *buf, size_t len,loff_t * off);
static ssize_t  my_write(struct file *filp, const char *buf, size_t len, loff_t * off);
static long       my_ioctl(struct file *filp, unsigned int cmd, unsigned long arg);

// структура ядра содержащая указатели на реализации операций для нашего девайса
static struct file_operations fops = {
   .owner        = THIS_MODULE,
   .read         = my_read,
   .write        = my_write,
   .open         = my_open,
   .release      = my_release,
   .unlocked_ioctl = my_ioctl,
};

static int my_open(struct inode *inode, struct file *file)
```

```
{
	print_msg("Driver Open Function Called...!!!\n");
	return 0;
}

static int my_release(struct inode *inode, struct file *file)
{
	print_msg("Driver Release Function Called...!!!\n");
	return 0;
}

static ssize_t my_read(struct file *filp, char __user *buf, size_t len, loff_t *off)
{
	uint8_t data[BUF_LEN] = {0};
	struct timespec t;
	print_msg("Driver Read Function Called...!!!\n");

	// функция для получния времени
	getnstimeofday(&t);
	snprintf(data, BUF_LEN, "Cur time(sec: %i, mircosec: %i)", (int)t.tv_sec, (int)t.tv_nsec);

	if (len > BUF_LEN) {
		len = BUF_LEN;
	}

	// функция для безхопасного копиования данных из области ядра в область
пользователя
	if (copy_to_user(buf, data, len)) {
		return -EFAULT;
	}

	return len;
}

static ssize_t my_write(struct file *filp, const char __user *buf, size_t len, loff_t *off)
{
	print_msg("Driver Write Function Called...!!!\n");
	return len;
}

static long my_ioctl(struct file *filp, unsigned int cmd, unsigned long arg) {
	print_msg("Driver Ioctl Function Called...!!!\n");
	return 0;
}

static int __init cdev_module_init(void)
{
	long res = 0;
	// выделяем область chardev и назначаем Major номер
	if((res = alloc_chrdev_region(&devt, 0, 1, "my_cdev")) < 0){
		print_msg("Cannot allocate major number\n");
		goto alloc_err;
```

```
	}
	print_msg("Major = %d Minor = %d\n", MAJOR(devt), MINOR(devt));

	// инициализируем новое устройство
	cdev_init(&my_cdev, &fops);
	// добавляем устройство в систему
	if((res = cdev_add(&my_cdev, devt, 1)) < 0){
		print_msg("Cannot add the device to the system\n");
		goto cdev_add_err;
	}

	// создаем класс sysfs
	dev_class = class_create(THIS_MODULE, "my_class");
	// обработка ошибок
	if(IS_ERR(dev_class)){
		res = PTR_ERR(dev_class);
		print_msg("Cannot create the struct class\n");
		goto class_err;
	}

	// создаем узел устройства /dev/my_cdev
	dev = device_create(dev_class, NULL, devt, NULL, "my_cdev");
	// обработка ошибок
	if(IS_ERR(dev)){
		res = PTR_ERR(dev);
		print_msg("Cannot create the Device\n");
		goto dev_create_err;
	}

	print_msg("Device Driver Insert...Done!!!\n");
	return 0;
dev_create_err:
	class_destroy(dev_class);
class_err:
cdev_add_err:
	unregister_chrdev_region(devt, 1);
alloc_err:
	return res;
}

static void __exit cdev_module_exit(void)
{
	device_destroy(dev_class, devt);
	class_destroy(dev_class);
	cdev_del(&my_cdev);
	unregister_chrdev_region(devt, 1);
	print_msg("Device Driver Remove...Done!!!\n");
}
module_init(cdev_module_init);
module_exit(cdev_module_exit);
```

Makefile модуля ядра.

```
ifneq ($(KERNELRELEASE),)
# kbuild part of makefile
obj-m := cdev_module.o
else
# normal makefile
KDIR ?= /lib/modules/`uname -r`/build

default:
	$(MAKE) -C $(KDIR) M=$$PWD
clean:
	$(MAKE) -C $(KDIR) M=$$PWD clean
endif
```

Програма для роботи з модулем ядра

```
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
#include<unistd.h>
#include<sys/ioctl.h>

#define BUF_LEN 256

int main(void) {
   int fd = open("/dev/my_cdev", O_RDONLY);

   if (fd < 0) {
      printf("Error number %d", errno);
   }

   unsigned char buf[BUF_LEN] = {0};
   read(fd, buf, BUF_LEN);

   printf("%s\n", buf);

   close(fd);
}
```