Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №3. Использование механизма виртуальной памяти в программах для ОС GNU/Linux
Тема

Преподаватель _____ А. С. Кузнецов
подпись, дата — инициалы, фамилия

Студент КИ19-17/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2021

1 Цель работы

Цель состоит в изучении принципов управления областями виртуальной памяти в ОС GNU/Linux.

2 Задачи

Выполнение работы сводится к следующим задачам.

1. Ознакомление с теоретическим материалом по управлению областями виртуальной памяти в ОС GNU/Linux.

2. Разработка программы в соответствии с полученным заданием, в которой должен использоваться механизм управления кучами либо стеком.

3. Написание настоящего отчета защита его с исходными текстами и исполняемым модулем программы. Исходные тексты программ должны содержать комментарии в стиле системы doxygen.

Требуется разработать программу в виде Linux-приложения, позволяющую манипулировать многоэлементными абстрактными структурами данных. Обязательны к реализации функции по добавлению одной структуры, модификации структуры (значения всех полей или только части из них), удаления структуры, чтения одной структуры, отображения содержимого всех структур (или части). Должен использоваться интерфейс командной строки (CLI). Описание элемента структуры данных и два дополнительных запроса данных, обязательных к реализации, даются индивидуально и приводятся далее.

Вариант 17. Структура данных: архипелаг; количество островов; количество обитаемых островов. Создать два запроса, позволяющих определить, имеются ли архипелаги, состоящие только из необитаемых островов, и получить список архипелагов с указанием количества островов в них. Программа принимает от пользователя беззнаковое целое десятичное число N – основание системы счисления ($N > 1$ и $N \leq 20$) и последовательности цифр в соответствии с заданной системой счисления. Затем программа выводит число на экран, переводит его в десятичную систему, выводит на экран, осуществляет его реверс,

выводит на экран значение измененной последовательности, переводит его в десятичную систему и выводит на экран.

3 Описание использованных при выполнении задания функций Linux API управления областями виртуальной памяти

3.1 void* malloc(size_t size);

malloc() распределяет size байтов и возвращает указатель на распределенную память. Память при этом не "очищается". При успешном вызове malloc() выделяет size байт памяти и возвращает указатель в начальную точку только что выделенной области. Содержимое памяти не определено, поэтому мы не должны рассчитывать, что память будет заполнена нулями. При ошибке malloc() возвращает NULL, а значение глобальной переменной errno устанавливается значение ENOMEM.

3.2 void free(void* ptr);

free() освобождает место в памяти, на которое указывает ptr, возвращенный, по-видимому, предшествующим вызовом функций malloc(), calloc() или realloc(). Иначе (или если уже вызывался free(ptr)) дальнейший ход событий непредсказуем. Если ptr равен NULL, то не выполняется никаких действий. Этот вызов освобождает память, на которую указывает ptr. Параметр ptr должен предварительно получен через malloc(), calloc() или realloc(). Это означает, что с помощью free() мы не сможем освобождать произвольные блоки памяти. Так, не получится освободить половину области памяти, передав указатель на ее середину. Такое действие приведет к появлению неопределенной памяти, что проявится в виде сбоя (ошибки сегментации).

4 Примеры использования этих функций в представленном программном коде

Листинг 1 – Код функции создания архипелага с использованием malloc()

```
/*! \brief Adds islands group in list by all its params
 *
 *  \details Checks if there is no any island group that has same name, or
 *  information about islands is incorrect. Creates a new IslandGroup and
 *  adds link with such content to the list if it's correct or returns NULL.
 *  Possible memory leak: allocates memory for new IslandsGroup and Link
 *  containing it.
 *
 *  \param objectList Pointer to any link of list to add new IslandGroup.
 *  \param name Name of new island group.
 *  \param islands Overall number of islands in new islands group.
 *  \param inhabitedIslands Number of inhabited islands in new islands group.
 *
 *  \return Pointer to added link or NULL if not added
 */ListLink* AddIslandGroup(ListLink* objectList, char* name, int islands,
                          int inhabitedIslands)
{
    if (FindIslandGroupLinkByName(objectList, name) != NULL ||
        islands < 1 || inhabitedIslands > islands)
    {
        return NULL;
    }
    IslandGroup* newIslandGroup;
    newIslandGroup = (IslandGroup*) malloc(sizeof(IslandGroup));
    newIslandGroup->name = (char*) malloc(sizeof(name));
    stpcpy(newIslandGroup->name, name);
    newIslandGroup->islands = islands;
    newIslandGroup->inhabitedIslands = inhabitedIslands;
    return ListLinkAdd(objectList, newIslandGroup);
}
```

Листинг 2 – Код функции удаления элемента двусвязного списка с использованием free()

```
/*! \brief Deletes element of linked list by pointer
 *
 *  \details Calls freeContent() of link to delete for freeing a content, then
 *  frees link itself.
 *
 *  \param linkToDelete Pointer to the link to delete
 *  \param freeContent Function to free content of linked list
 *
 *  \return Head of a new list
 */
ListLink* ListLinkDelete(ListLink* linkToDelete, void* (* freeContent)(void*))
{
    ListLink* linkToReturn;
    if (linkToDelete->next != NULL)
    {
        linkToDelete->next->previous = linkToDelete->previous;
    }
```

Окончание листинга 2

```
    linkToReturn = ListLinkHead(linkToDelete);
    if (linkToReturn != linkToDelete)
    {
        linkToDelete->previous->next = linkToDelete->next;
    }
    else
    {
        linkToReturn = linkToDelete->next;
    }
    freeContent(linkToDelete->content);
    free(linkToDelete);
    return linkToReturn;
}
```

5 Тестовые примеры работы программы

На рисунках далее приведены примеры работы программы.

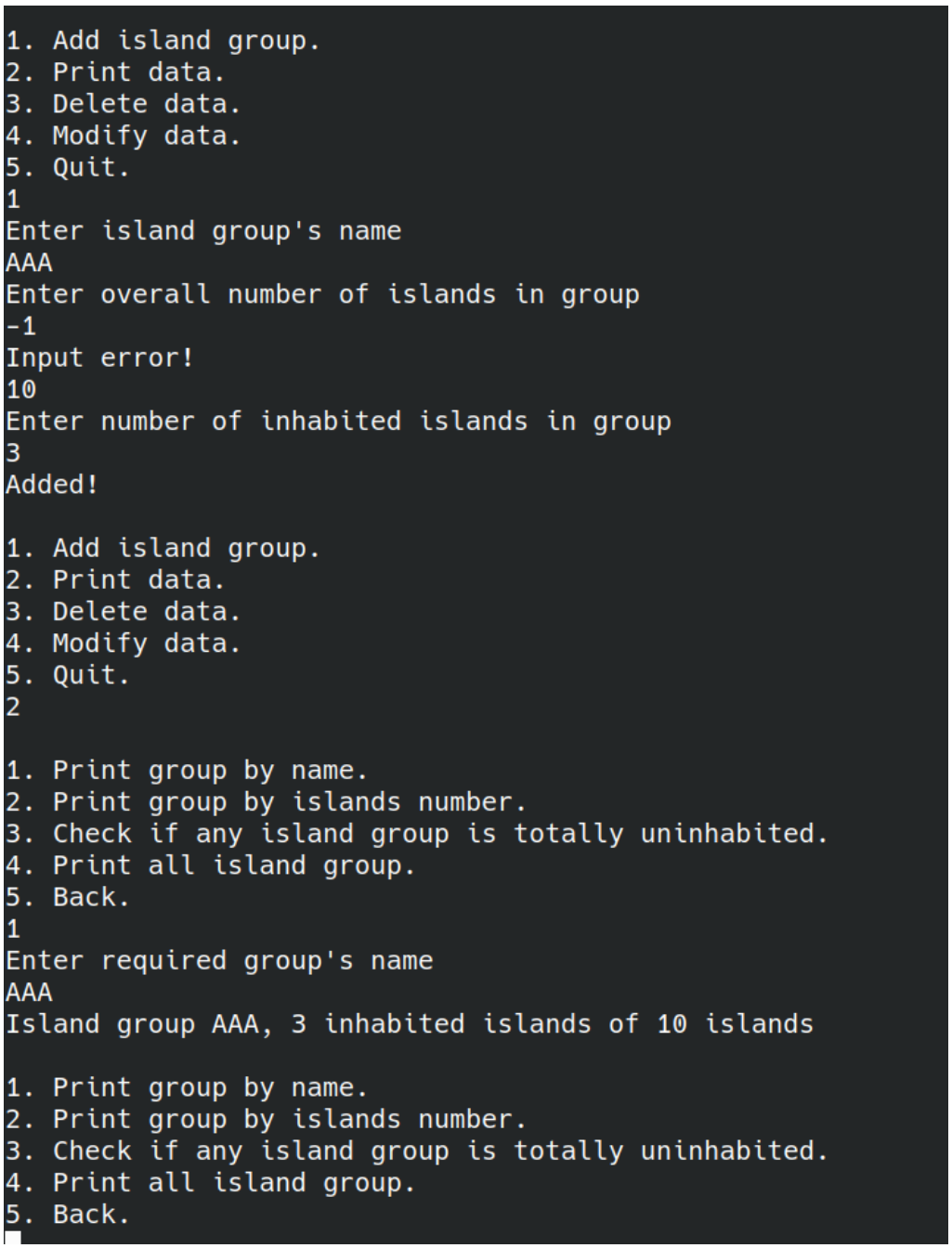

Рисунок 1 – Пример работы программы 1, добавление архипелага

```
1. Add island group.
2. Print data.
3. Delete data.
4. Modify data.
5. Quit.
1
Enter island group's name
BBB
Enter overall number of islands in group
8
Enter number of inhabited islands in group
12
There is a conflict in islands numbers or group with such name already exists. None added

1. Add island group.
2. Print data.
3. Delete data.
4. Modify data.
5. Quit.
`
Input error!
1
Enter island group's name
BBB
Enter overall number of islands in group
10
Enter number of inhabited islands in group
0
Added!
```

Рисунок 2 – Пример работы программы 2, ввод противоречивых данных

```
1. Change island group name.
2. Change island group overall number of islands.
3. Change island group number of inhabited islands.
4. Back.
1
Enter required group's name
AAA
Enter new name for this island group
newAAA
Name changed!
1. Add island group.
2. Print data.
3. Delete data.
4. Modify data.
5. Quit.
2

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
4
Island group newAAA, 2 inhabited islands of 2 islands

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
```

Рисунок 3 – Пример работы программы 3, переименование архипелага

```
4. Print all island group.
5. Back.
4
Island group AAA, 4 inhabited islands of 12 islands
Island group BBB, 0 inhabited islands of 8 islands
Island group CCC, 2 inhabited islands of 10 islands
Island group DDD, 0 inhabited islands of 10 islands

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
2
Enter required number of islands
10
Island group CCC, 2 inhabited islands of 10 islands
Island group DDD, 0 inhabited islands of 10 islands

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
3
Found at least one totally uninhabited island group
```

Рисунок 4 – Пример работы программы 4, запросы в соответствии с заданием

```
Enter name of the group to delete
DDD
Deleted!
1. Add island group.
2. Print data.
3. Delete data.
4. Modify data.
5. Quit.
3
Enter name of the group to delete
BBB
Deleted!
1. Add island group.
2. Print data.
3. Delete data.
4. Modify data.
5. Quit.
2

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
2
Enter required number of islands
10
Island group CCC, 2 inhabited islands of 10 islands

1. Print group by name.
2. Print group by islands number.
3. Check if any island group is totally uninhabited.
4. Print all island group.
5. Back.
3
There is no any uninhabited island groups
```

Рисунок 5 – Пример работы программы 5, удаление архипелагов