НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №4
З дисципліни
«Операційні системи»

Виконав:
студент групи КА-65
Іванов Д.С.
Перевірив:
Коваленко А.Є.

Київ 2017

ВИКОРИСТАННЯ ПРОГРАМНИХ ЗАСОБІВ ОПЕРАЦІЙНИХ СИСТЕМ.

Мета. Основні цілі роботи:

1. навчитись застосовувати стандартні програми, пакети і утиліти операційної системи.
2. отримати досвід програмування в Ubuntu мовою C, C++
3. ознайомитись з особливостями створення макросів
4. отримати досвід програмування системних викликів сучасних операційних систем

Завдання:

1. Скласти на мові C в каталозі bin для виконання дій згідно із завданням. Програма повинна передбачати виведення повідмлення за навності помилки.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[])
{
   char buff[2048];
   //Открытие файла
   int file2;
   int file = open("1.txt", O_RDWR);
   if(file == -1){
         printf("Файла не существует!");
       }
   //Вывод содержимого
   read(file, buff, 2048);
   printf("Содержимое файла:%s", buff);
   //Обнуляем буфер
   memset(buff, 0, 2048);
   printf("    Введите строку, которую следует дописать в файл  ");
   //Считывание строки
   scanf("%s", buff);
   //Смещение
   if(lseek(file, 0, SEEK_SET)==-1){
   printf(" lseek error! ");
   };
   //Запись в файл
   write(file, buff, strlen(buff));
   close(file);
   return 0;
}
```

2. Виконати компіляцію програми

daniel@daniel-Aspire-E3-112 ~/Documents/univer/OS/mine/laba4 $ **gcc laba4-with.c**

3. Розробити контрольний приклад

daniel@daniel-Aspire-E3-112 ~/Documents/univer/OS/mine/laba4 $ **./a.out**
Содержимое файла:000000 Введите строку, которую следует дописать в файл
111

daniel@daniel-Aspire-E3-112 ~/Documents/univer/OS/mine/laba4 $ **./a.out Содержимое файла:111000 Введите строку, которую следует дописать в файл**

4. Надати відкомпільованому файлу права на виконання і прописати шлях до bin

```
daniel@daniel-Aspire-E3-112 ~/ivanov/bin $ chmod a+x laba4-with.c
daniel@daniel-Aspire-E3-112 ~/ivanov/bin $ PATH=$PATH:~/ivanov/bin/
```

Прототип *lseek*:

```
off_t lseek(int fd, off_t offset, int whence);
```

```
1 off_t lseek(int fd, off_t offset, int whence);
```

Аргумент *fd* — дескриптор файла, *offset* — смещение, *whence* — тип смещения. Возвращает количество байтов, на которое сместился указатель от начала файла, если не было ошибки, иначе -1. В качестве *whence* можно ипспользовать следующие макросы:

- *SEEK_SET* — смещение в файле устанавливается в значение offset;
- *SEEK_CUR* — смещение в файле устанавливается в текущее значение + offset;
- *SEEK_END* — смещение в файле устанавливается в конец файла + offset.

Кроме смещения текущей позиции, системный вызов *lseek* может послужить в других целях. Например для определения текущей позиции:

```
if((fd=open(argv[1], O_RDONLY | O_BINARY))==-1) { /* открытие для записи */
printf("Cannot open file.\n");
exit(0);
}int pos = lseek(file, 0, SEEK_CUR);
```

```
1 int pos = lseek(file, 0, SEEK_CUR);
```

Или например для определения размера файла:

```
int size = lseek(file, 0, SEEK_END);
```

```
1 int size = lseek(file, 0, SEEK_END);
```

Контрольні питання по роботі:

1. Пояснити особливості застосування С, С++.

В операційних системах сімейства UNIX застосовують стандартну мову програмування ANSI C (1983 р.). Компілятор С відноситься до процедурних мов третього покоління. Мова С++ відноситься до об′єктно - орієнтованих мов програмування і є підмножиною С. Для трансляції вихідних текстів програм на С, С++ часто використовуються програми – транслятори GNU gcc, g++ відповідно.
Для розробки засобів керування процесами і ресурсами використовують системні каталоги і каталоги користувачів.
Мова С++ являє собою об'ктно –орієнтованим розширенням мови С і широко застосовується при створенні графічних додатків.. Компілятор С++ коректно компілює програми на мові С. Функції, які викликаються, можуть автоматично розрізнятись залежно від типів даних, що називається перевантаженням імен функції.
Для встановлення режимів роботи компілятора застосовують перемикачі у командному рядку перемикачі компілятора GNU C/C++.

2. Які системні виклики застосовують при керуванні процесами ?

Системні виклики є функціями (наприклад, fork, exec, wait), тіло яких розміщено у резидентному ядрі ОС. Системний виклик проходить у два етапи – виклику програми на асемблері з генерацією програмного переривання і реакції ядра на виклик. Реакція ядра відбувається у такій послідовності
1. Перехід до привілейованого режиму.
2. Аналіз джерела - процесу запиту, і підключення u-area цього процесу
до адресного простору ядра (context switching).
3. Отримання аргументів з пам`яті процесу, який зробив запит.
4. Визначення потрібних дій ядра (зокрема, номера системного виклику).
5. Перевірка коректності решти аргументів виклику.
6. Перевірка прав процесу на припустимість зробленого запиту.
7. Виклик тіла системного виклику (з можливим засинанням процесу).
8. Повернення відповіді у пам`ять процесу.
9. Відключення привілейованого режиму.
10. Повернення з переривання.
Системний виклик зазвичай повертає ознаку успішного виконання (0), невдачі (-1) або містовне значення (наприклад, дескриптор файлу під час виконання open()). У разі невдалого завершення змінна errno набуває значення номера помилки (зазвичай коди помилок описано в include-файлі <errno.h>).
За системним викликом fork() створюється копія процесу, який надіслав цей запит. Значенням fork є 0 для нового процесу і pid нового процесу у вихідному.
За системного виклику exec(), який заміняє програму на нову успадковану процесом, усі відкриті канали також успадковуються (не закриваються). Ця команда має кілька різновидів. Наприклад, для виклику a.out можна скористатись двома з них.
char *path;
char *argv[], *envp[], *arg0, ..., *argn;
execle(path, arg0, arg1, ..., argn, NULL, envp);
execve(path, argv, envp);
При цьому програма, яка виконується цим процесом, замінюється на програму з файлу path.

3. Пояснити особливості створення макросів

Функції і макрокоманди дозволяють об'єднати групи виразів і операторів для сумісного використання. Розробка власних функцій дозволяє розширити бібліотечні функції С або удосконалити їх для забезпечення більш ефективнго програмування і обробки даних.
В ANSI C виділяють оголошення, виклик і опис функції. Макрос дозволяє замінити елементарні значення довільним текстом.