|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ**

*к лабораторной работе №2*

*По курсу: «Операционные системы»*

*На тему: «Загружаемые модули ядра»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

# Задание

1. Реализовать загружаемый модуль ядра, который при загрузке записывает в системный журнал информацию о процессах. О каждом процессе записать: название, идентификатор, название предка, идентификатор предка. Модуль должен собираться при помощи make файла.

# Листинг

|  |
| --- |
| #include <linux/module.h> // макросы MODULE\_<...> |

|  |
| --- |
| #include <linux/kernel.h> // функции ядра |

|  |
| --- |
| #include <linux/init.h> // \_\_init и \_\_exit |

|  |
| --- |
| #include <linux/sched.h> // task\_struct - планировщик |

#include <linux/init\_task.h> // next\_task()

#define FTW\_F 1 // файл, не являющийся каталогом

#define FTW\_D 2 // каталог

#define FTW\_DNR 3 // каталог, недоступный для чтения

#define FTW\_NS 4 // файл, информацию о котором нельзя получить с помощью stat

// Тип функции, которая будет вызываться для каждого встреченного файла

**typedef** **int** **Handler**(**const** **char** \*,**const** **struct** stat \*, **int**);

**static** **Handler** counter;

**static** **int** dopath(**const** **char** \*filename, **int** depth, **Handler** \*);

**static** **long** nreg, ndir, nblk, nchr, nfifo, nslink, nsock, nTotal;

**int** **main**(**int** argc, **char** \* argv[])

{

**int** ret = -1;

**if** (argc != 2)

{

printf("ERROR: wrong arguments.\nUse: ./app <dir>\n");

exit(-1);

}

ret = dopath(argv[1], 0, counter); //выполняет всю работу

nTotal = nreg + ndir + nblk + nchr + nfifo + nslink + nsock;

**if** (nTotal == 0)

nTotal = 1; // во избежание деления на 0

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nSummary:\n\n");

printf("Обычные файлы:\t%7ld, %5.2f %%\n", nreg, nreg\*100.0/nTotal);

printf("Каталоги:\t%7ld, %5.2f %%\n", ndir, ndir\*100.0/nTotal);

printf("Специальные файлы блочных утсройств:\t%7ld, %5.2f %%\n", nblk, nblk\*100.0/nTotal);

printf("Специальные файлы символьных утсройств:\t%7ld, %5.2f %%\n", nchr, nchr\*100.0/nTotal);

printf("FIFO:\t\t%7ld, %5.2f %%\n", nfifo, nfifo\*100.0/nTotal);

printf("Символические ссылки:\t%7ld, %5.2f %%\n", nslink, nslink\*100.0/nTotal);

printf("Сокеты:\t%7ld, %5.2f %%\n\n", nsock, nsock\*100.0/nTotal);

printf("Всего:\t%7ld\n", nTotal);

exit(ret);

}

// Обход дерева каталогов

**static** **int** dopath(**const** **char** \*filename, **int** depth, **Handler** \*func)

{

**struct** stat statbuf;

**struct** dirent \* dirp;

DIR \*dp;

**int** ret = 0;

**if** (lstat(filename, &statbuf) < 0) // ошибка

**return**(func(filename, &statbuf, FTW\_NS));

**for** (**int** i = 0; i < depth; ++i)

printf("|\t");

**if** (S\_ISDIR(statbuf.st\_mode) == 0) // не каталог

**return**(func(filename, &statbuf, FTW\_F)); // отобразить в дереве

**if** ((ret = func(filename, &statbuf, FTW\_D)) != 0)

**return**(ret);

**if** ((dp = opendir(filename)) == NULL) // каталог недоступен

**return**(func(filename, &statbuf, FTW\_DNR));

chdir(filename);

**while** ((dirp = readdir(dp)) != NULL && ret == 0)

{

**if** (strcmp(dirp->d\_name, ".") != 0 &&

strcmp(dirp->d\_name, "..") != 0 ) // пропуск каталогов . и ..

{

ret = dopath(dirp->d\_name, depth + 1, func);

}

}

chdir("..");

**if** (closedir(dp) < 0)

perror("Can’t close dir");

**return**(ret);

}

**static** **int** counter(**const** **char** \*pathame, **const** **struct** stat \*statptr, **int** type)

{

**switch**(type)

{

**case** FTW\_F:

printf( "-- %s\n", pathame);

**switch**(statptr->st\_mode & S\_IFMT)

{

**case** S\_IFREG: nreg++; **break**;

**case** S\_IFBLK: nblk++; **break**;

**case** S\_IFCHR: nchr++; **break**;

**case** S\_IFIFO: nfifo++; **break**;

**case** S\_IFLNK: nslink++; **break**;

**case** S\_IFSOCK: nsock++; **break**;

**case** S\_IFDIR:

perror("Dir has type FTW\_F");

**return**(-1);

}

**break**;

**case** FTW\_D:

printf( "-- %s/\n", pathame);

ndir++;

**break**;

**case** FTW\_DNR:

perror("One of dirs is closed.");

**return**(-1);

**case** FTW\_NS:

perror("Error in stat.");

**return**(-1);

**default**:

perror("Unknown file type.");

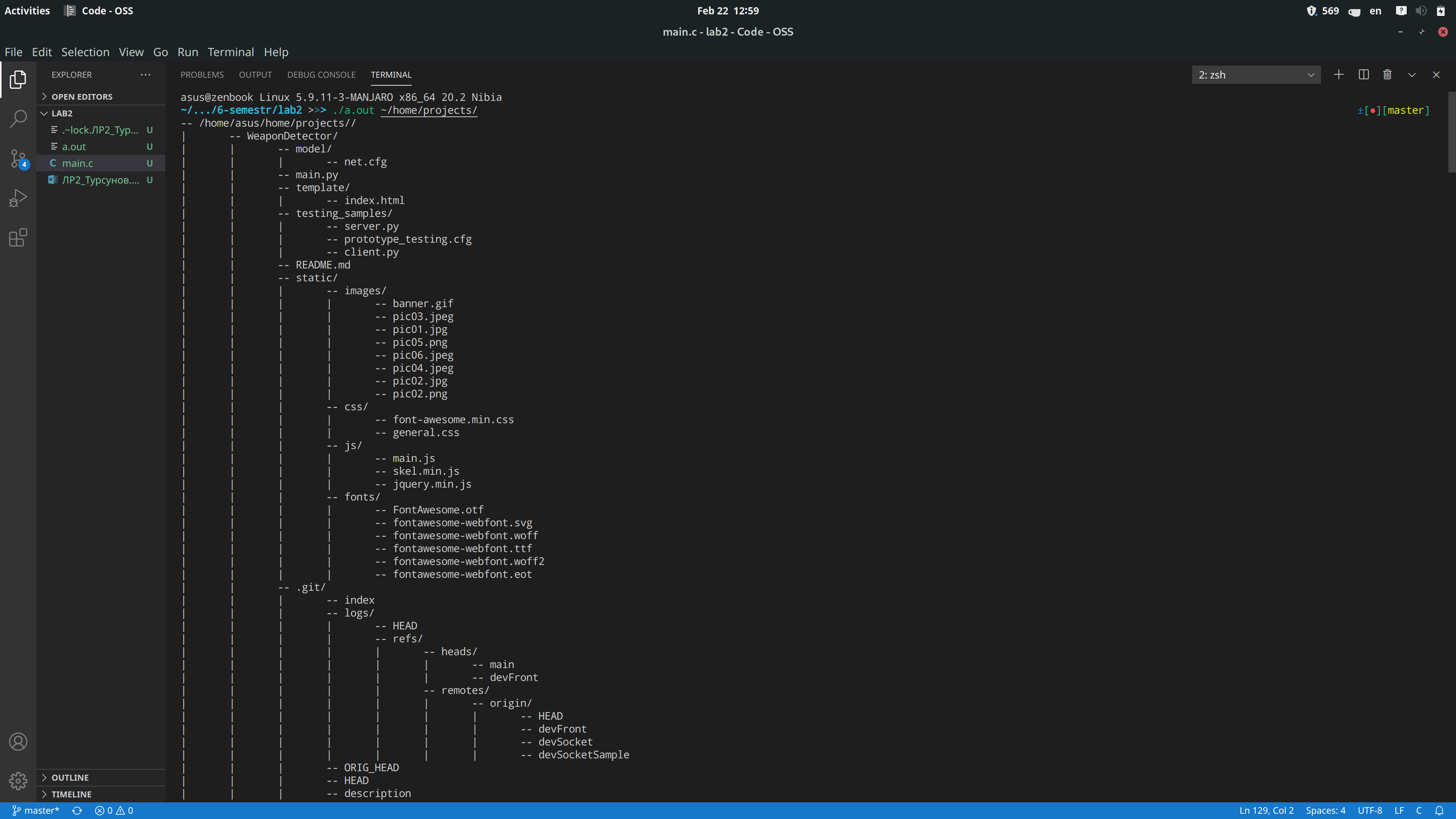
**return**(-1);

}

**return**(0);

}

# Результат работы программы

****

