

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Шебзухова Татьяна Александровна
Должность: Директор Пятигорского института (филиал) Северо-Кавказского
федерального университета
Дата подписания: 23.08.2023 14:49:46
Уникальный программный ключ:
d74ce93cd40e39275c3ba2f58486412a1c8ef96f

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по выполнению лабораторных работ

по дисциплине

«Управление информационными ресурсами обработки цифрового контента»

для направления подготовки **09.03.02 Информационные системы и
технологии**

направленность (профиль) **Информационные системы и технологии
обработки цифрового контента**

**Пятигорск
2022**

**ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ**

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

ВВЕДЕНИЕ

Управление информационными ресурсами обработки цифрового контента – дисциплина, изучающая основные принципы организации и методы управления аппаратными и программными средствами в вычислительных машинах (ВМ) и системах (ВС). В результате изучения курса студенты должны хорошо ориентироваться в принципах построения современных ВМ и ВС, архитектурных решениях, направленных на повышение производительности вычислительных машин, областях применения машин и систем с различной архитектурой и направлениях их развития.

Цели дисциплины:

- ознакомление студентов с основными принципами организации аппаратного обеспечения ЭВМ и систем, принципами работы периферийных устройств и их взаимодействия в составе системы.
- подготовка специалиста с современными знаниями об электронно-вычислительных машинах, высокой квалификации с широким теоретическим кругозором, способным осваивать новое в науке и технике.

Задачами дисциплины «Управление информационными ресурсами обработки цифрового контента» являются:

- 1) формирование комплекса знаний, умений и навыков, связанных с применением средств современной вычислительной техники, необходимых для правильного использования электронно-вычислительных машин и их модернизации;
- 2) формирование набора общепрофессиональных и профессиональных компетенций.

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

Лабораторная работа 1

Управление системными ресурсами средствами shell-интерпретатора

Цель работы: познакомиться с основами программирования на уровне командного языка Shell путём написания Shell-программ для работы с файловой системой.

Теоретическое обоснование

Обычно в ОС UNIX доступны несколько интерпретаторов. Наиболее распространены Bourne-shell (или просто - shell), C-shell, Korn-shell. В идейном плане все эти интерпретаторы близки и в дальнейшем речь будет идти о стандартном Shell (/bin/sh).

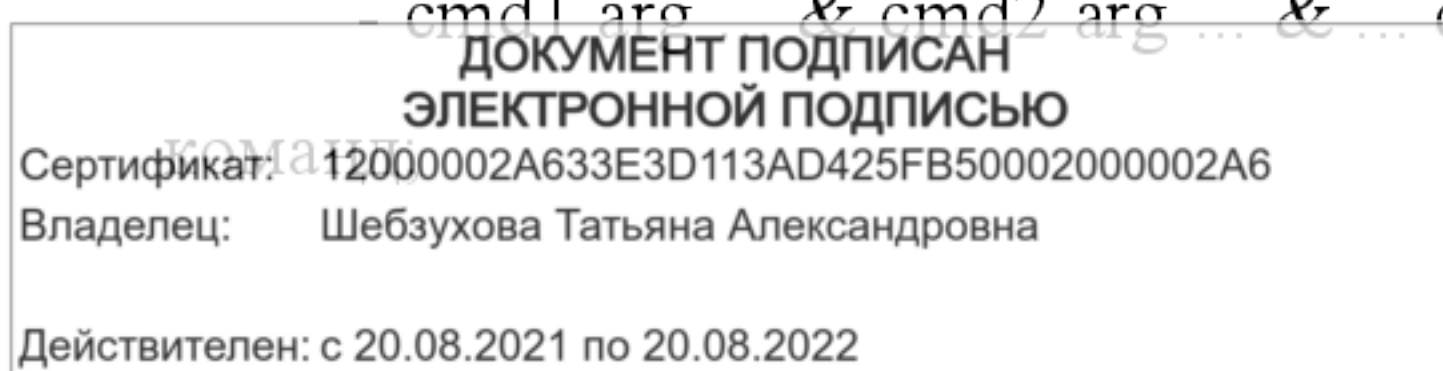
Работая на командном языке, пользователь может вводить переменные, присваивать им значения, выполнять простые команды, строить составные команды, управлять потоком выполнения команд, объединять последовательность команд в процедуры (командные файлы). На уровне командного языка доступны такие свойства системы как соединение процессов через программный канал, направление стандартного ввода/вывода в конкретные файлы, синхронное и асинхронное выполнение команд.

Если указанный интерпретатору файл является текстовым и содержит команды командного языка (командный файл) и при этом имеет разрешение на выполнение (помечен "x"), Shell-интерпретатор интерпретирует и выполняет команды этого файла. Другой способ вызова командного файла - использование команды sh (вызов интерпретатора), в котором первым аргументом указывается имя командного файла.

Коротко перечислим средства группирования команд и перенаправления ввода/вывода:

- `cmd1 arg ...; cmd2 arg ...; ... cmdN arg ...` – последовательное выполнение команд;

- `cmd1 arg ... & cmd2 arg ... & ... cmdN arg ...` – асинхронное выполнение команд;



- `cmd1 arg ... && cmd2 arg ...` – зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала нулевое значение;

- `cmd1 arg ... || cmd2 arg ...` – зависимость последующей команды от предыдущей таким образом, что последующая команда выполняется, если предыдущая выдала ненулевое значение;

- `cmd > file` – стандартный вывод направлен в файл `file`;

- `cmd >> file` – стандартный вывод направлен в конец файла `file`;

- `cmd < file` – стандартный ввод выполняется из файла `file`;

- `cmd1 | cmd2` – конвейер команд, в котором стандартный вывод команды `cmd1` направлен на стандартный вход команды `cmd2`.

Shell-переменные могут хранить строки текста. Правила формирования их имен аналогичны правилам задания имен переменных в обычных языках программирования. При необходимости присвоить Shell-переменной значение, содержащее пробелы и другие специальные знаки, оно заключается в кавычки. При использовании Shell-переменной в выражении ее имени должен предшествовать знак `$`. В последовательности символов те из них, которые составляют имя, должны быть выделены в `{ }` или `" "`. Кроме того, интерпретатор Shell автоматически присваивает значения пяти своим переменным:

- `$?` – значение, возвращаемое последней выполняемой командой;

- `$$` – идентификационный номер процесса Shell;

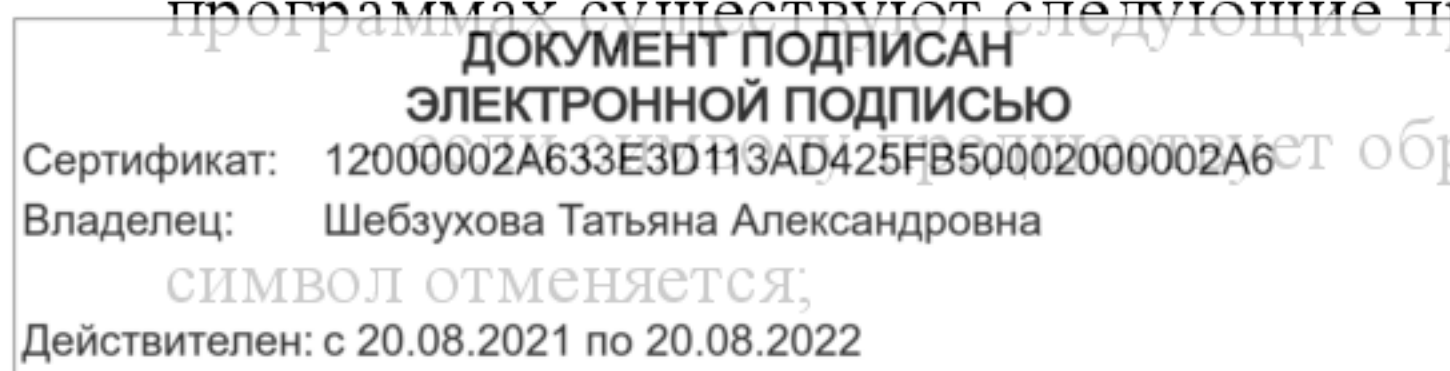
- `$!` – идентификационный номер фонового процесса, запускаемого интерпретатором Shell последним;

- `$#` – число аргументов, переданных в Shell;

- `$–` – флаги, переданные в Shell.

Для отмены специальных символов (`$`, `|`, пробел и т.д.) в Shell-

программах существуют следующие правила:



Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

СИМВОЛ ОТМЕНЯЕТСЯ;

Действителен: с 20.08.2021 по 20.08.2022

- отменяется специальный смысл всех символов, вошедших в последовательность, заключенную в апострофы.

При вызове Shell-программ им могут передаваться параметры. Соответствующие аргументы в Shell-программах идентифицируются \$1, \$2, \$3 и т.д. Кроме того, переменная \$0 соответствует имени выполняемой Shell-программы, а переменная \$# - числу аргументов в команде.

Shell-интерпретатор дает возможность выполнять подстановку результатов выполнения команд в Shell-программах. Если команда заключена в одиночные обратные кавычки, то интерпретатор Shell выполняет эту команду и подставляет вместо нее полученный результат.

Наиболее важные команды для составления Shell-программ:

- команда echo выводит в выходной поток значения своих аргументов;
- команда expr выполняет арифметические действия над своими аргументами;
- команда eval обеспечивает дополнительный уровень подстановки своих аргументов, а затем их выполнение;
- команда test с соответствующими ключами проверяет необходимое условие;
- команда sleep служит для реализации задержки.

Программные конструкции Shell-программ:

Условный оператор if:

```
if if_list
  [then then_list
  elif elif_list]
  then then_list
[else else_list]
fi
```

Цикл for

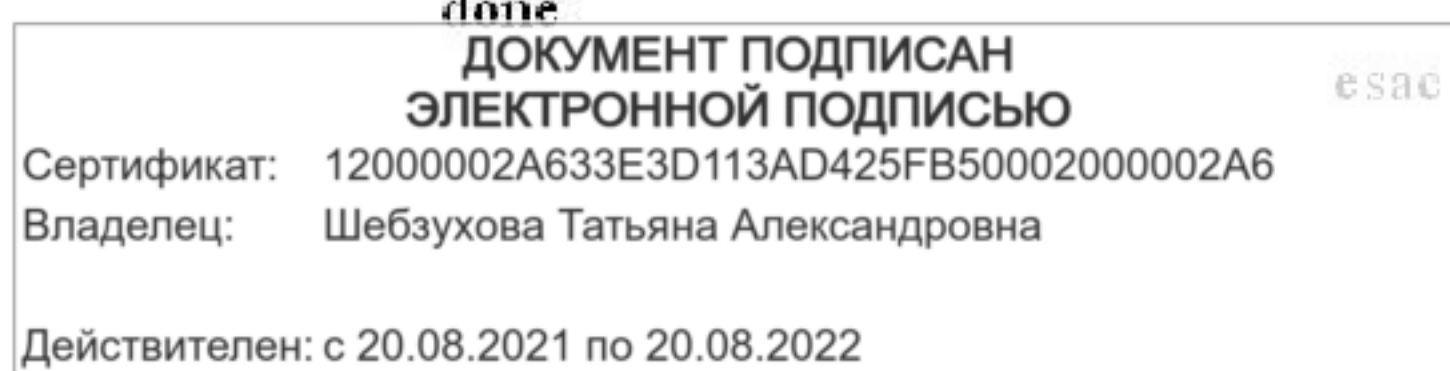
```
for name [in word1, word2...]
do do_list
done
```

Циклы while и until:

```
while while_list
do do_list
done
until until_list
do do_list
done
```

Структура case

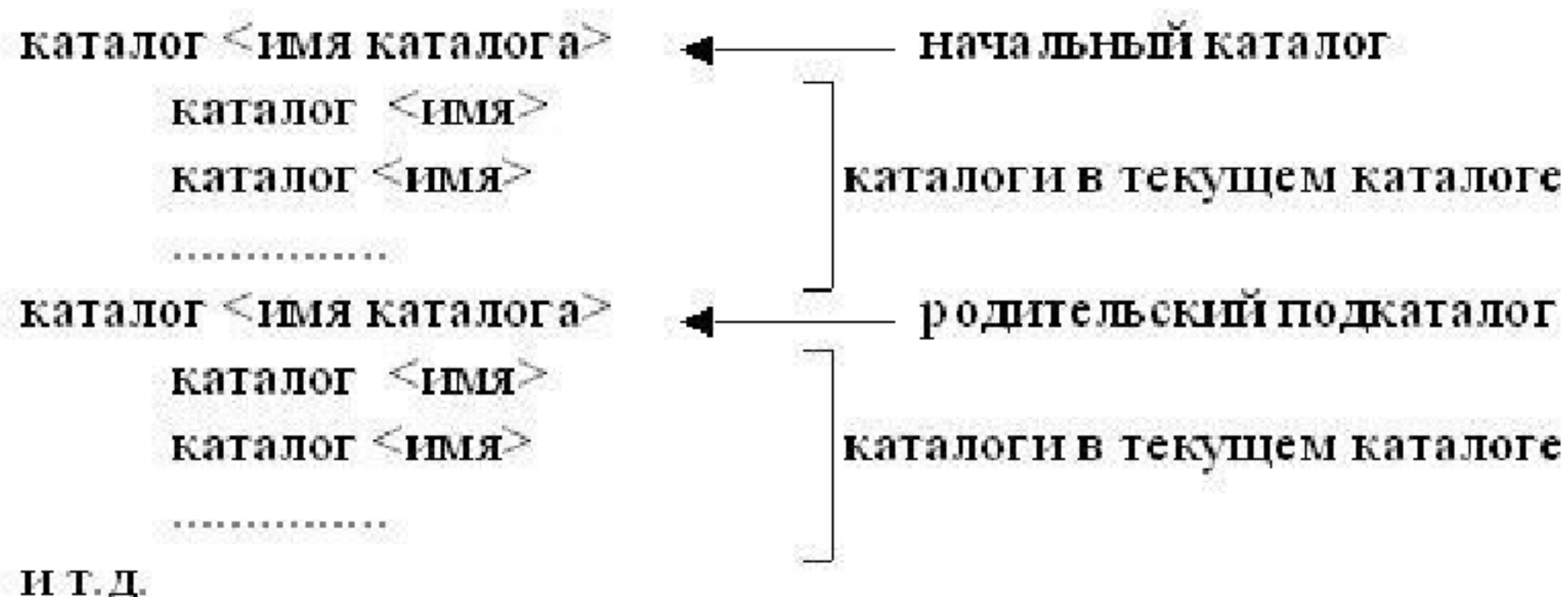
```
case word in
  pattern1) part_list;;
  pattern2) part_list;;
```



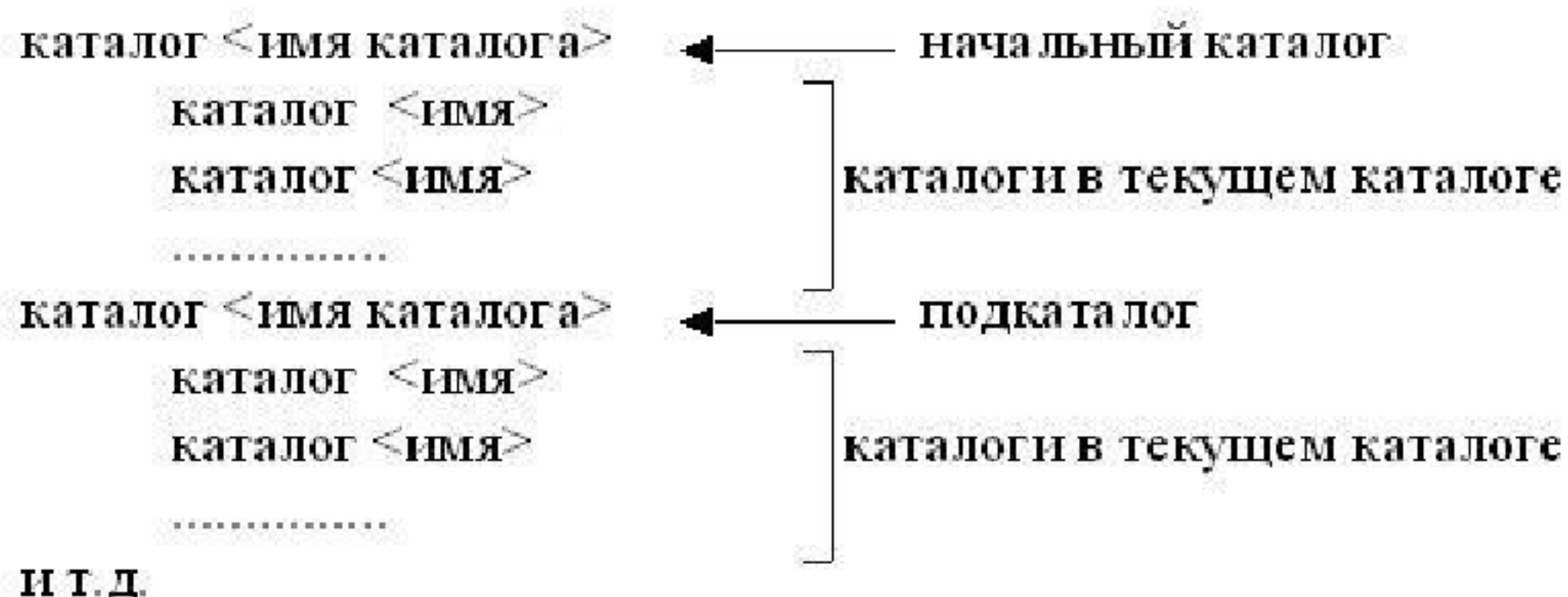
Варианты заданий к лабораторной работе

1. Shell-программа выводит имена тех каталогов в каталоге, которые в себе содержат каталоги. Имя каталога задано параметром Shell-программы.

2. Shell-программа просматривает каталог, имя которого указано параметром Shell-программы и выводит имена встретившихся каталогов. Затем осуществляет переход в родительский каталог, который становится текущим и повторяются указанные действия до тех пор, пока текущим каталогом не станет корневой каталог. Форма вывода результата:

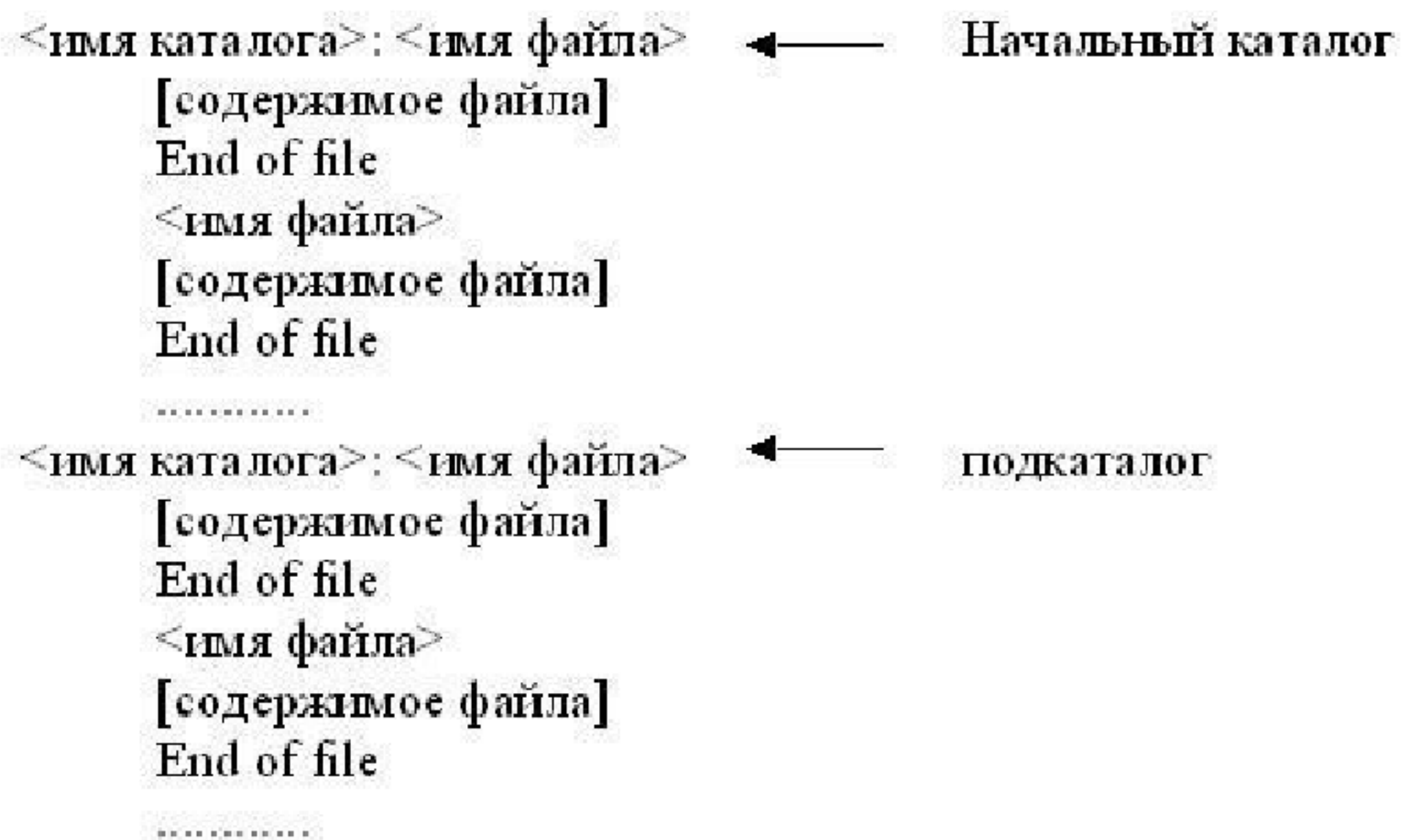


3. Shell-программа подсчитывает количество и выводит перечень каталогов в хронологическом порядке (по дате создания) в поддереве, начиная с каталога, имя которого задано параметром Shell-программы. Форма вывода результата:

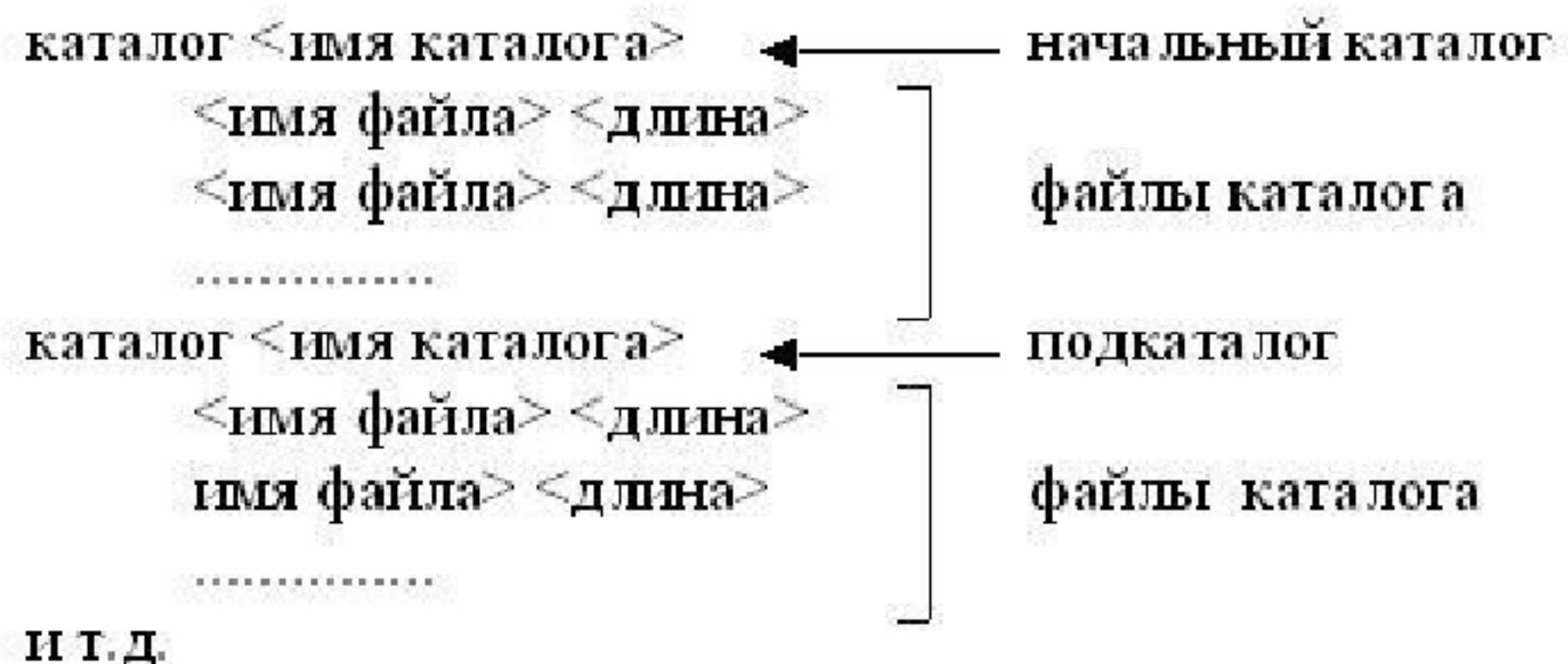


3. Shell-программа объединяет все временные файлы с указанным суффиксом (например, .tmp) в поддереве, начиная с каталога, имя которого задано параметром Shell-программы. Результат объединения помещается либо в указанный Shell-программой файл, либо выводится на экран в форме:

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат: 12000002A633E3D113AD425FB50002000002A6
Владелец: Шебзухова Татьяна Александровна
Действителен: с 20.08.2021 по 20.08.2022

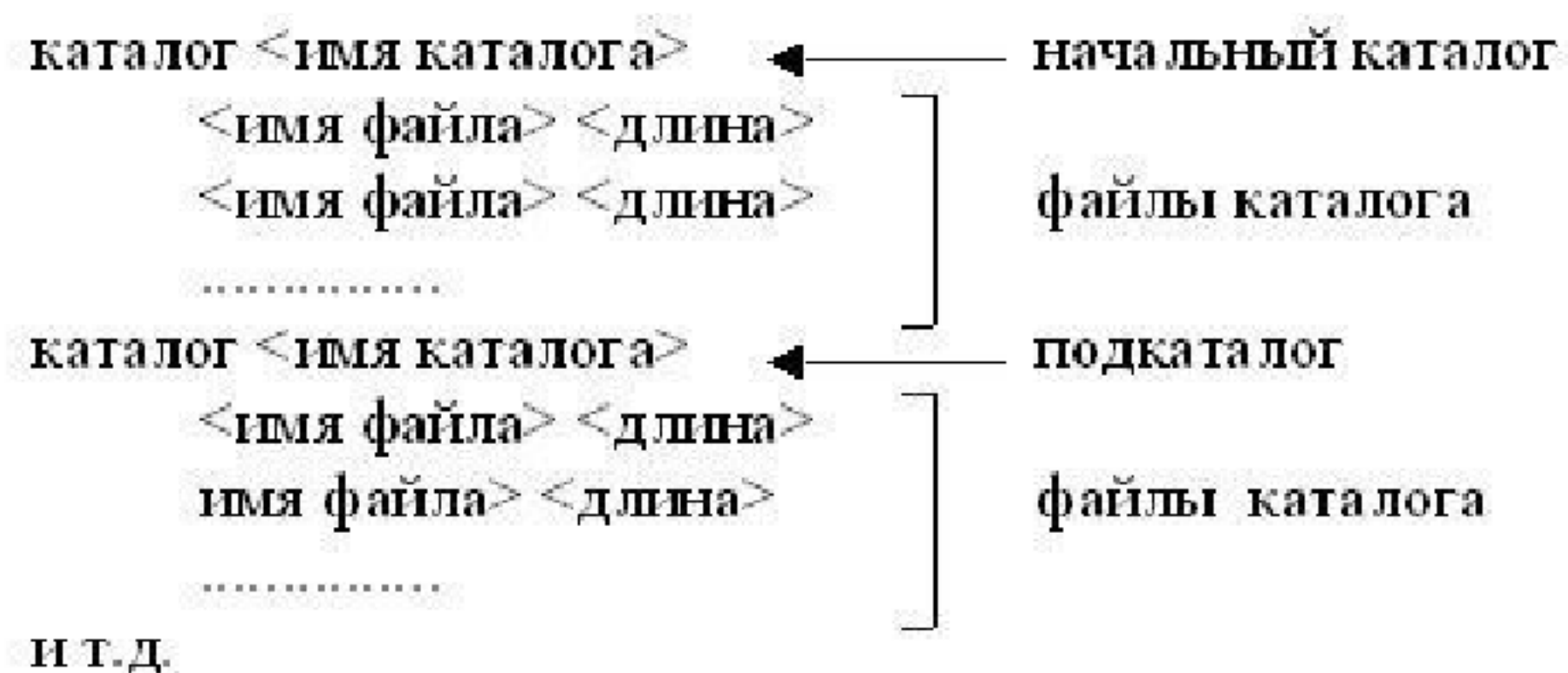


4. Shell-программа периодически с некоторым интервалом удаляет все временные файлы с указанным суффиксом (например, .tmp) в поддереве, начиная с каталога, имя которого задано параметром Shell-программы и выводит при этом список объединенных файлов в форме:

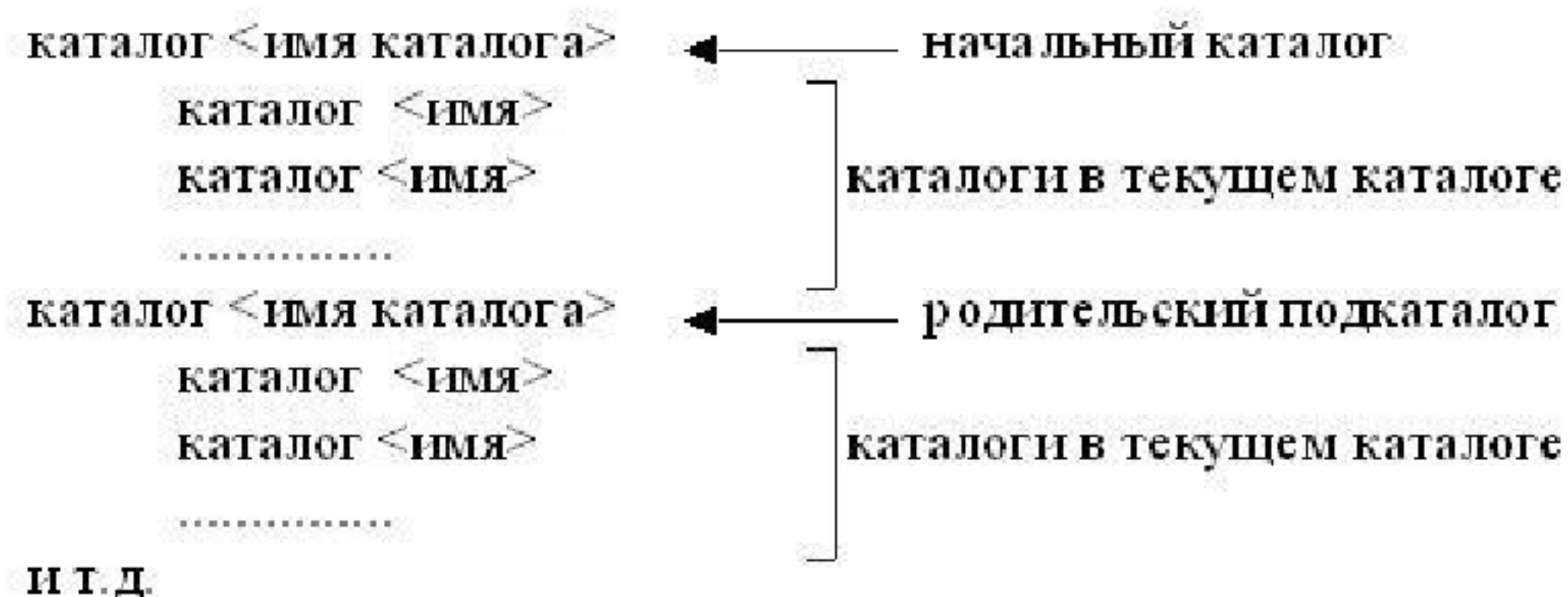


5. Shell-программа выводит содержимое каталога, имя которого указано параметром Shell-программы. При выводе сначала перечисляются имена каталогов, а затем в алфавитном порядке имена файлов с указанием их длин, даты создания и числа ссылок на них.

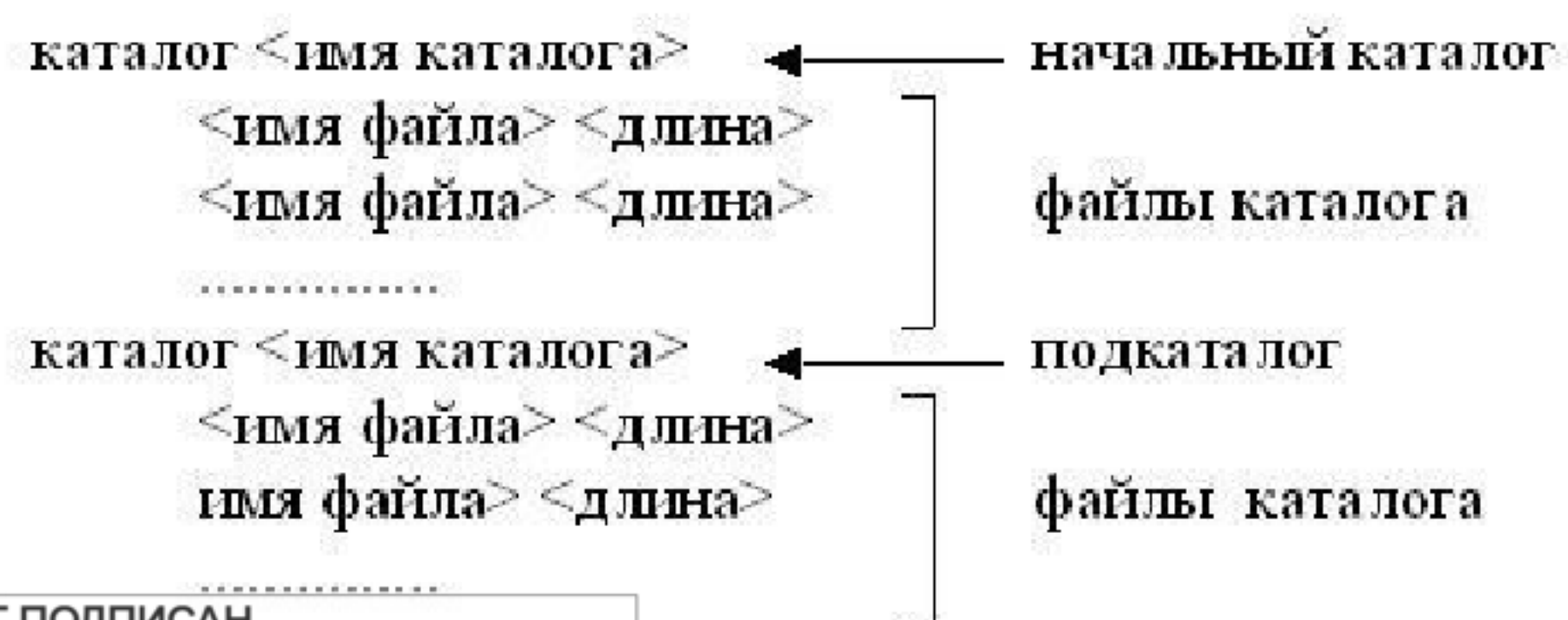
6. Shell-программа подсчитывает количество и выводит список всех файлов (без каталогов) в порядке уменьшения их длин в поддереве, начиная с каталога, имя которого задано параметром Shell-программы. Форма вывода



7. Shell-программа просматривает каталог, имя которого указано параметром Shell-программы и выводит имена встретившихся файлов. Затем осуществляет переход в родительский каталог, который становится текущим и повторяются указанные действия до тех пор, пока текущим каталогом не станет корневой каталог. Форма вывода результата:



8. Shell-программа подсчитывает количество и выводит список всех файлов (без каталогов) в алфавитном порядке в поддереве, начиная с каталога, имя которого задано параметром Shell-программы. Форма вывода результата:



9. Shell-программа выводит имена тех каталогов в каталоге, которые в себе не содержат каталогов. Имя каталога задано параметром Shell-программы.

Контрольные вопросы

1. Что такое внутренние и внешние команды Shell-интерпретатора? Приведите примеры внутренних команд.
2. Какие существуют средства группирования команд? Приведите примеры использования.
3. Как осуществляется перенаправление ввода-вывода?
4. В чем сущность конвейера команд? Приведите примеры использования.
5. Как средствами Shell выполнить арифметические действия над Shell-переменной?
6. Каковы правила генерации имен файлов?
7. Как выполняется подстановка результатов выполнения команд?
8. Как интерпретировать строку `cmd1 & cmd2 & ?`
9. Как интерпретировать строку `cmd1 && cmd2 & ?`
10. Как интерпретировать строку `cmd1 || cmd2 & ?`
11. В каком режиме выполняется интерпретатор команд Shell?
12. Кем и в каком режиме осуществляется чтение потока символов с терминала интерпретатором Shell?

Лабораторная работа 2

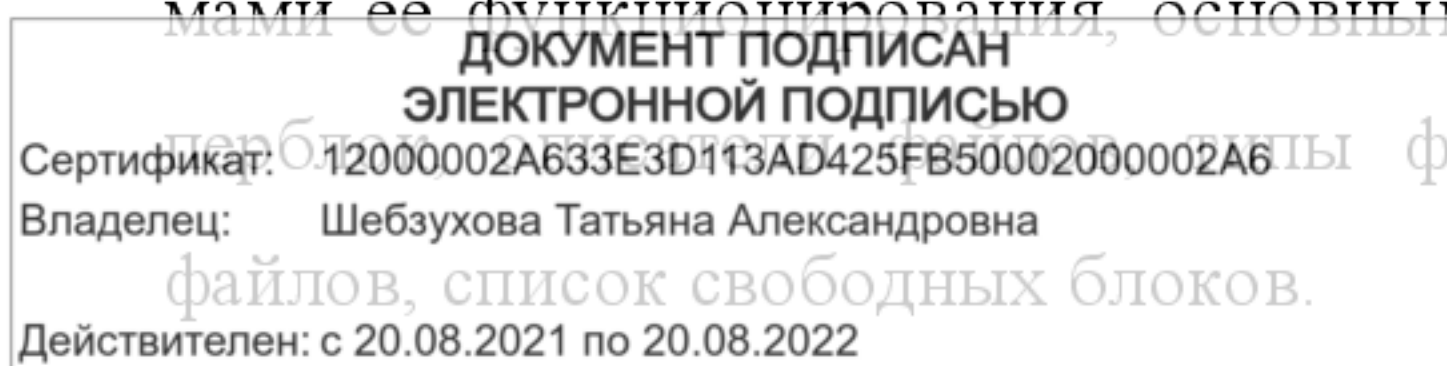
Файловая система ОС UNIX

Цель работы: ознакомиться с файловой системой ОС UNIX, механиз-

мами ее функционирования, основными элементами файловой системы: су-

пербном, вторичном, третичном файлах, списке свободных описателей

файлов, список свободных блоков.



Теоретическое обоснование

Интерфейс между пользовательской программой и внешним устройством (или между двумя пользовательскими программами) в ОС UNIX осуществляется в рамках единой структуры данных, называемой файлом ОС UNIX.

Всякий файл ОС UNIX в соответствие с его типом может быть отнесен к одной из следующих четырех групп: обычные файлы, каталоги, специальные файлы, каналы.

Обычный файл представляет собой совокупность блоков диска, входящих в состав файловой системы ОС UNIX. В указанных блоках может быть произвольная информация.

Каталоги представляют собой файлы особого типа, отличающиеся от обычных, прежде всего, тем, что осуществить запись в них может только ядро ОС UNIX, в то время как доступ по чтению может получить любой пользовательский процесс, имеющий соответствующие полномочия. Каждый элемент каталога состоит из двух полей: поля имени файла и поля, содержащего указатель на описатель файла, где хранится вся информация о файле: дата создания, размер, код защиты, имя владельца и т.д. В любом каталоге содержится, по крайней мере, два элемента, содержащие в поле имени файла имена "." и "..". Элемент каталога, содержащий в поле имени файла контекст ".", в поле ссылки содержит ссылку на описатель файла, описывающий этот каталог. Элемент каталога, содержащий в поле имени файла контекст "..", в поле ссылки содержит ссылку на описатель файла, в котором хранится информация о родительском каталоге текущего каталога.

Специальные файлы - это некоторые файлы, каждому из которых ставится в соответствие свое внешнее устройство, поддерживаемое ОС UNIX и имеющее специальную структуру. Его нельзя использовать для хранения данных, как обычный файл или каталог. В то же время над специальным

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат: 12000002A633E3D113AD425FB50002000002A6
Владелец: Шебзухова Татьяна Александровна
Действителен: с 20.08.2021 по 20.08.2022

файлом можно выполнять те же операции, что и над обычным файлом: открывать, вводить и/или выводить информацию и т.д. Результат применения

любой из этих операций зависит от того, какому конкретному устройству соответствует обрабатываемый специальный файл, однако в любом случае будет осуществлена соответствующая операция ввода-вывода на внешнее устройство, которому соответствует выбранный специальный файл.

Четвертый вид файлов - каналы - будет рассмотрен отдельно в последующих лабораторных работах.

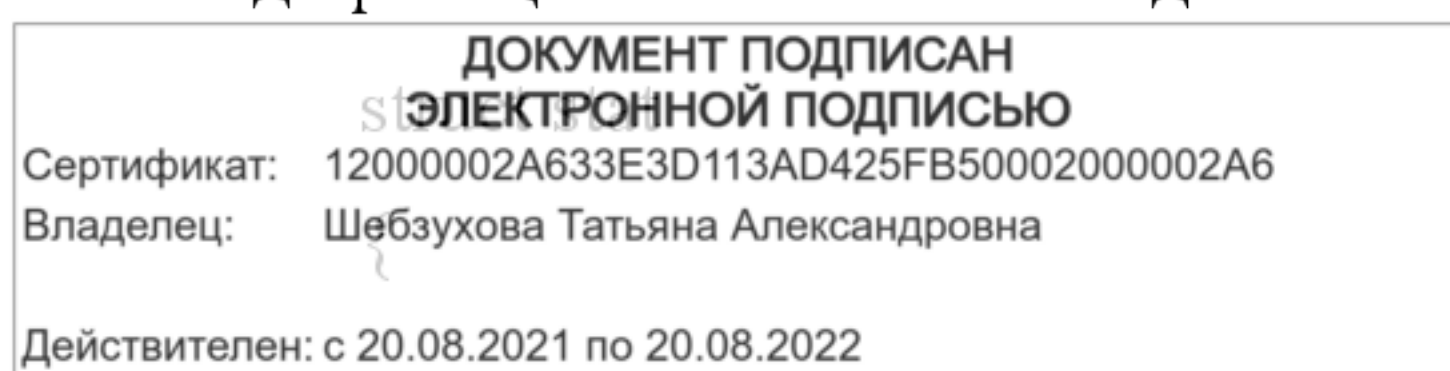
Системные функции ОС UNIX для работы с файловой системой:

Возвращают дескриптор файла	open, creat, dup, pipe, close
Преобразуют имя в описатель	open, creat, chdir, chmod, stat, mkfifo, mount, mknod, link, unmount, unlink, chown
Назначают inode	creat, link, unlink, mknod
Работают с атрибутами	chown, chmod, stat
Ввод/вывод из файла	read, write, lseek
Работают со структурой файловой системы	mount, unmount

Для получения информации о типе файла необходимо воспользоваться системными вызовами stat() (fstat()). Формат системных вызовов stat() (fstat()):

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *name, struct stat *stbuf);
int fstat(int fd, struct stat *stbuf);
```

Оба системных вызова помещают информацию о файле (в первом случае специфицированном именем name, а во втором - дескриптором файла fd) в структурную переменную, на которую указывает stbuf. Вызывающая функция должна позаботиться о резервировании места для возвращаемой информации; в случае успеха возвращается 0, в противном случае -1 и код ошибки errno. Описание структуры stat содержится в файле sys/stat.h. С небольшими модификациями она имеет вид:



```

dev_t st_dev; /* device file */
ino_t st_ino; /* file serial inode */
ushort st_mode; /* file mode */
short st_nlink; /* number of links */
ushort st_uid; /* user ID */
ushort st_gid; /* group ID */
dev_t st_rdev; /* device ident */
off_t st_size; /* size of file */
time_t st_atime; /* last access time */
time_t st_mtime; /* last modify time */
time_t st_ctime; /* last status change */
};

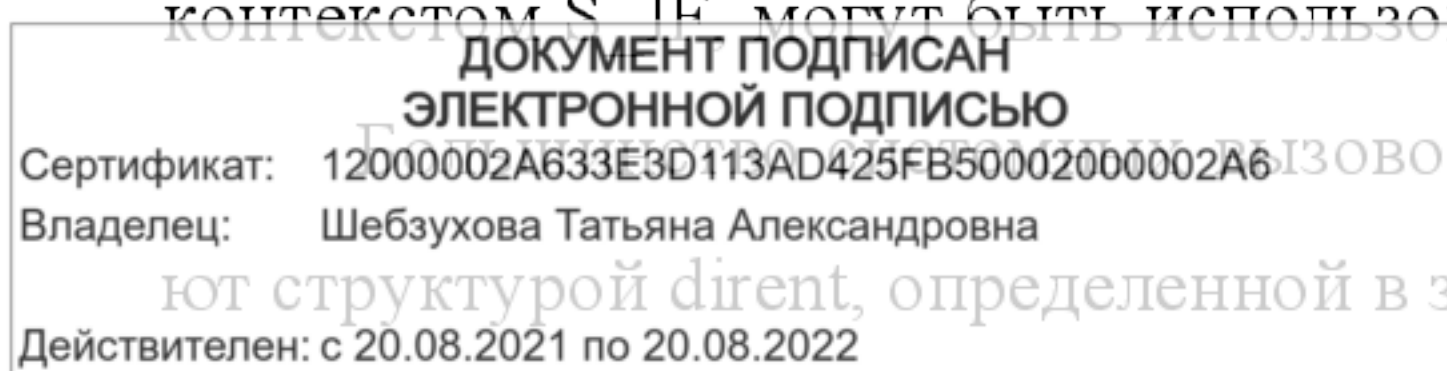
```

Поле `st_mode` содержит флаги, описывающие файл. Флаги несут следующую информацию:

`S_IFMT 0170000` – тип файла
`S_IFDIR 0040000` – каталог
`S_IFCHR 0020000` – байт-ориентированный специальный файл
`S_IFBLK 0060000` – блок-ориентированный специальный файл
`S_IFREG 0100000` – обычный файл
`S_IFFIFO 0010000` – дисциплина FIFO
`S_ISUID 04000` – идентификатор владельца
`S_ISGID 02000` – идентификатор группы
`S_ISVTX 01000` – сохранить свопируемый текст
`S_ISREAD 00400` – владельцу разрешено чтение
`S_IWRITE 00200` – владельцу разрешена запись
`S_IXEC 00100` – владельцу разрешено выполнение

Символьные константы, четыре первых символа которых совпадают с контекстом `S_IF`, могут быть использованы для определения типа файла.

Флаги, начинающиеся с `S_`, работающих с каталогами, оперируют структурой `dirent`, определенной в заголовочном файле `<dirent.h>`.



```

struct dirent
{
    ino_t d_ino;      /* номер индексного дескриптора */
    char d_name[DIRSIZ]; /* имя файла */
}

```

Создание и удаление каталога выполняется системными вызовами `mkdir()` и `rmdir()`. При создании каталога посредством системного вызова `mkdir()` в него помещаются две ссылки (". и "..").

```

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int mkdir (char *pathname, mode_t mode);
int rmdir (char *pathname);

```

Открытие и закрытие каталога выполняется системными вызовами `opendir()` и `closedir()`. При успешном открытии каталога системный вызов возвращает указатель на переменную типа `DIR`, являющуюся дескриптором каталога, определенную в файле `dirent.h` и используемую при чтении и записи в каталог. При неудачном вызове возвращается значение `NULL`.

```

#include <sys/types.h>
#include <dirent.h>

DIR *opendir (char *dirname);
int closedir (DIR *dirptr); /* dirptr - дескриптор каталога */

```

Для смены каталога служит системный вызов `chdir()`:

```

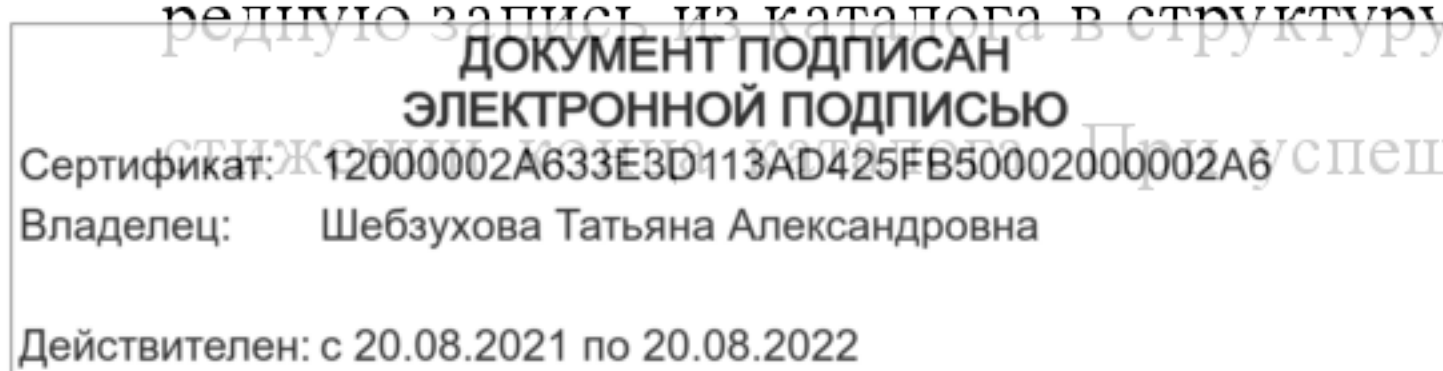
#include <unistd.h>

int chdir (char *pathname);

```

Чтение записей каталога выполняется системным вызовом `readdir()`. Системный вызов `readdir()` по номеру дескриптора каталога возвращает очередную запись из каталога в структуру `dirent`, либо нулевой указатель при до-

стижении каталога. При успешном чтении, указатель каталога пере-



мещается к следующей записи. Дополнительный системный вызов `rewinddir()` переводит указатель каталога к первой записи каталога.

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir (DIR *dirptr);
void rewinddir (DIR *dirptr);
```

Варианты заданий к лабораторной работе

1. Разработать программу, которая осуществляет просмотр текущего каталога и выводит на экран его содержимое группами в порядке возрастания числа ссылок на файлы (в том числе имена каталогов). Группа представляет собой объединение файлов с одинаковым числом ссылок на них.

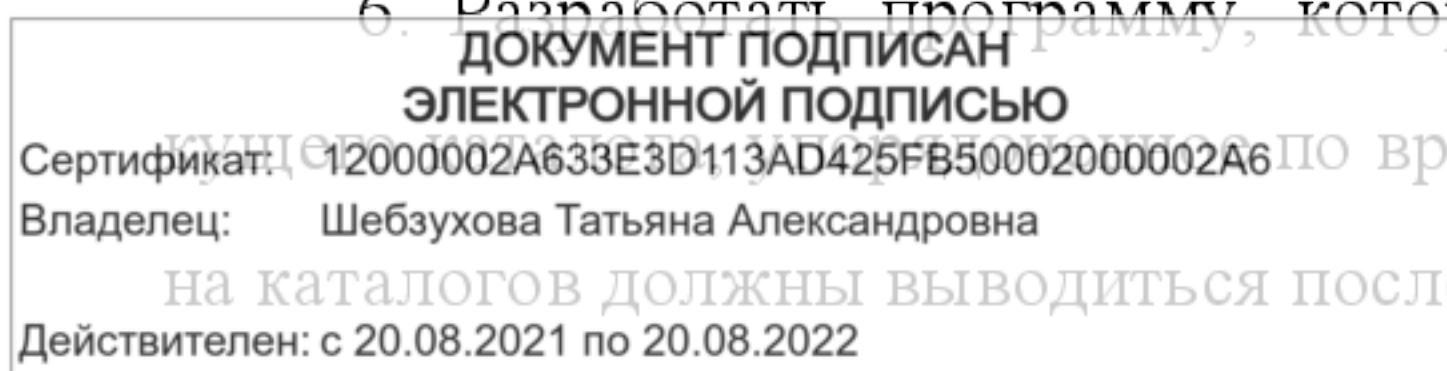
2. Разработать программу, которая просматривает текущий каталог и выводит на экран имена всех встретившихся в нем файлов с заданным расширением. Затем осуществляется переход в родительский каталог, который затем становится текущим, и указанные выше действия повторяются до тех пор, пока текущим каталогом не станет корневой каталог.

3. Разработать программу, которая просматривает текущий каталог и выводит на экран имена всех встретившихся в нем обычных файлов. Затем осуществляется переход в родительский каталог, который затем становится текущим, и указанные выше действия повторяются до тех пор, пока текущим каталогом не станет корневой каталог.

4. Разработать программу, которая выводит на экран имена тех каталогов, которые находятся в текущем каталоге и не содержат в себе подкаталогов.

5. Разработать программу, которая выводит на экран имена тех каталогов, которые находятся в текущем каталоге и содержат в себе подкаталоги.

6. Разработать программу, которая выводит на экран содержимое текущего каталога, а также имена каталогов, находящихся в нем. Имена каталогов должны выводиться последними.



7. Разработать программу, которая выводит на экран содержимое текущего каталога в порядке возрастания размеров файлов. При этом имена каталогов должны выводиться первыми.

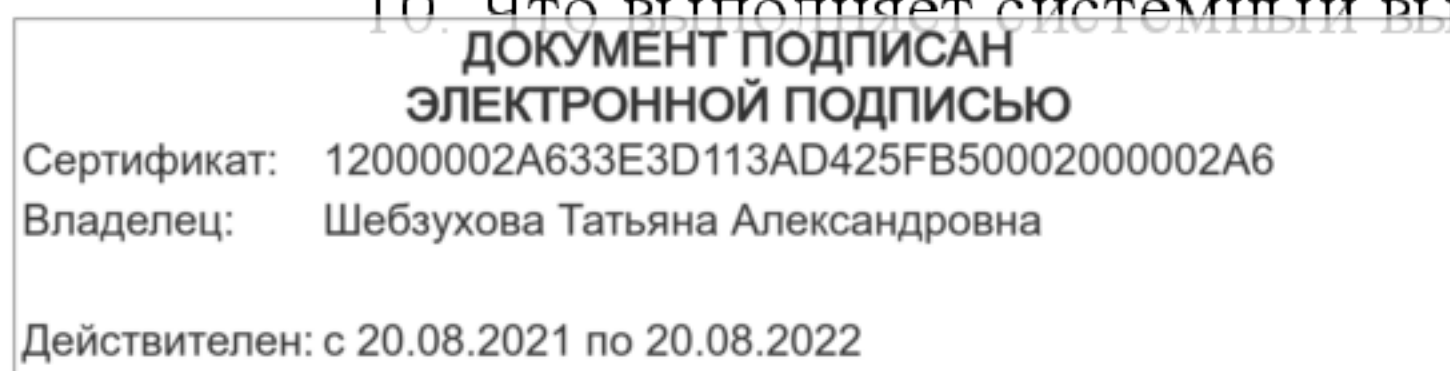
8. Разработать программу, которая выводит на экран содержимое текущего каталога в алфавитном порядке. Каталоги не выводить.

9. Разработать программу, которая просматривает текущий каталог и выводит на экран имена всех встретившихся в нем каталогов. Затем осуществляется переход в родительский каталог, который затем становится текущим, и указанные выше действия повторяются до тех пор, пока текущим каталогом не станет корневой каталог.

10. Разработать программу, которая осуществляет просмотр текущего каталога и выводит на экран имена находящихся в нём каталогов, упорядочив их по числу файлов и каталогов, содержащихся в отображаемом каталоге. Для каждого такого каталога указывается число содержащихся в нём файлов и каталогов.

Контрольные вопросы

1. Что представляет собой суперблок?
2. Что представляет собой список свободных блоков?
3. Что представляет собой список свободных описателей файлов?
4. Как производится выделение свободных блоков под файл?
5. Как производится освобождение блоков данных, занятых под файл?
6. Каким образом осуществляется монтирование дисковых устройств?
7. Каково назначение элементов структуры stat?
8. Каким образом осуществляется защита файлов в ОС UNIX?
9. Каковы права доступа к файлу, при которых владелец может выполнять все операции (r, w, x), а прочие пользователи - только читать?
10. Что выполняет системный вызов lseek(fd, (off_t)0, SEEK_END)?



Лабораторная работа 3

Структура системы управления вводом-выводом в ОС UNIX

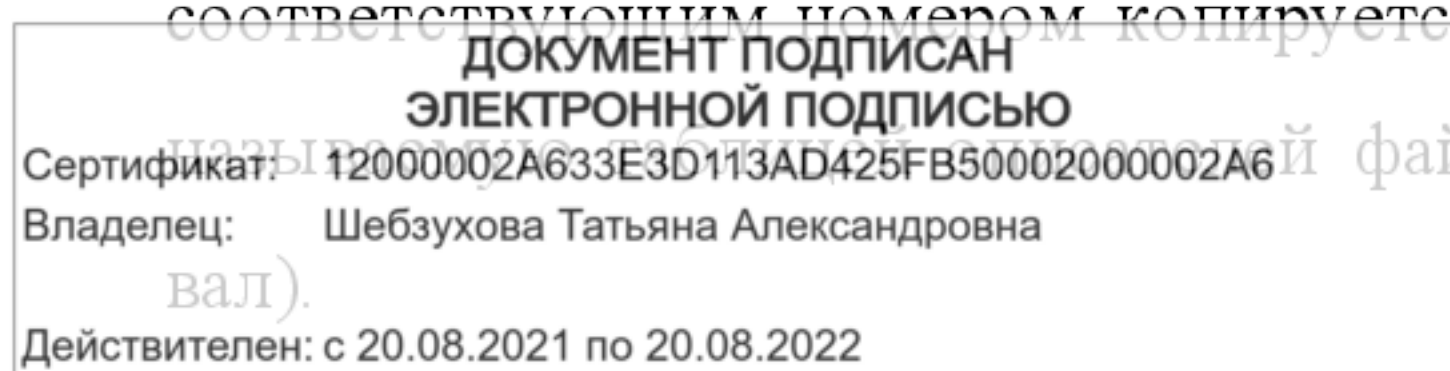
Цель работы: ознакомиться с системой управления вводом-выводом в ОС UNIX и основными структурами данных, используемыми этой системой; исследовать механизм работы системы управления вводом-выводом.

Теоретическое обоснование

Основным назначением системы управления вводом-выводом ОС UNIX является создание интерфейса между программой и внешним устройством компьютера. Поскольку любая операция ввода-вывода осуществляется как операция ввода-вывода в файл, то логическая структура программного интерфейса, реализуемого системой управления вводом-выводом, не зависит ни от типа данных, ни от типа внешнего устройства компьютера.

При осуществлении операций ввода-вывода в файл, специфицированный пользовательским дескриптором файла, ОС UNIX ставит в соответствие используемому системному вызову последовательность программных запросов к аппаратуре компьютера с помощью целого ряда связанных наборов данных, структура которых поддерживается самой ОС UNIX, ее файловой системой и системой управления вводом-выводом. Основным из упомянутых наборов можно считать таблицу описателей файлов.

Таблица описателей файлов представляет собой хранящуюся в оперативной памяти компьютера структуру данных, элементами которой являются копии описателей файлов, по одной на каждый файл ОС UNIX, к которому была осуществлена попытка доступа. При выполнении операции открытия файла в ОС UNIX сначала по полному имени файла определяется элемент каталога, где в поле имени содержится имя файла, для которого производится операция открытия файла. В найденном элементе каталога из поля ссылки извлекается порядковый номер описателя файла. Затем описатель файла с соответствующим номером копируется в оперативную память, в ее область,



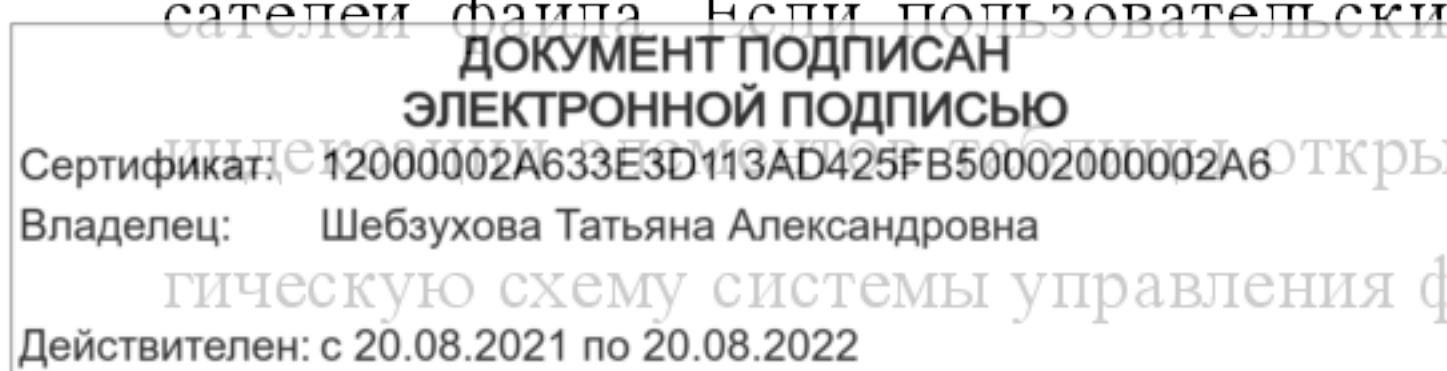
С таблицей описателей файлов тесно связана другая структура данных, называемая таблицей файлов. Каждый элемент таблицы файлов содержит информацию о режиме открытия файла, специфицированным при открытии файла, а также информацию о положении указателя чтения-записи. При каждом открытии файла в таблице файлов появляется новый элемент.

Один и тот же файл ОС UNIX может быть открыт несколькими не связанными друг с другом процессами, при этом ему будет соответствовать один элемент таблицы описателей файлов и столько элементов таблицы файлов, сколько раз этот файл был открыт. Однако из этого правила есть одно исключение: оно касается случая, когда файл, открытый процессом, потом открывается процессом-потомком, порожденным с помощью системного вызова `fork()`. При возникновении такой ситуации операции открытия файла, осуществленной процессом-потомком, будет поставлен в соответствие тот из существующих элементов таблицы файлов (в том числе положение указателя чтения-записи), который в свое время был поставлен в соответствие операции открытия этого файла, осуществленной процессом-предком.

Третий набор данных называется таблицей открытых файлов процесса. Каждому процессу в ОС UNIX сразу после порождения ставится в соответствие таблица открытых файлов процесса. Если, в свою очередь, указанный процесс порождает новый процесс, например, с помощью системного вызова `fork()`, то процессу-потомку ставится в соответствие таблица открытых файлов процесса, которая в первый момент функционирования процесса-потомка представляет собой копию таблицы открытых файлов процесса-предка.

В результате каждый элемент таблицы открытых файлов процесса содержит указатель местоположения соответствующего элемента таблицы файлов, которая в свою очередь, содержит ссылку на элемент таблицы описателей файла. Если пользовательский дескриптор файла использовать для

открытых файлов процесса, то получим логическую схему системы управления файлами (вводом-выводом).



Лабораторная работа предполагает написание программы, показывающей действия системы управления вводом-выводом при выполнении некоторых действий с файлами. Программа должна демонстрировать динамику формирования таблиц и их изменений в процессе указанных в варианте задания событий.

При этом при выполнении тех заданий, где требуется продемонстрировать создание таблиц описателей файлов, информацию о файле необходимо получать с помощью системных вызовов stat (fstat), поскольку именно информация, хранящаяся в описателе файла, в основном и помещается системным вызовом stat (fstat) в структуру, специфицированную его вторым выходным параметром.

Полученную информацию из структуры stat, дополненную именем файла и следует трактовать в качестве таблицы описателей файлов.

В тех заданиях, где требуется отслеживать динамику создания и модификации таблиц файлов и таблиц открытых файлов процесса, эти таблицы должны программно моделироваться при возникновении событий, указанных в заданиях лабораторной работы. Никаких действий по созданию процессов в программах выполнять не требуется.

Структура таблицы файлов в программах лабораторной работы (упрощенный вариант) должна иметь вид:

file name	number inode	locate point	Ссылка на таблицу описателей файлов (если необходимо)		
xxxxxx	xxxxxx	xxxxxx	xxxxxx	←	для каждого файла
xxxxxx	xxxxxx	xxxxxx	xxxxxx	←	
xxxxxx	xxxxxx	xxxxxx	xxxxxx	←	
xxxxxx	xxxxxx	xxxxxx	xxxxxx	←	

Структура таблицы открытых файлов в программах должна иметь вид:

закрытие пользовательского файла (моделирование `close(fd)`, где `fd` - дескриптор пользовательского файла).

После каждого из этапов печатаются таблица описателей файлов, таблица файлов, таблица открытых файлов процессов.

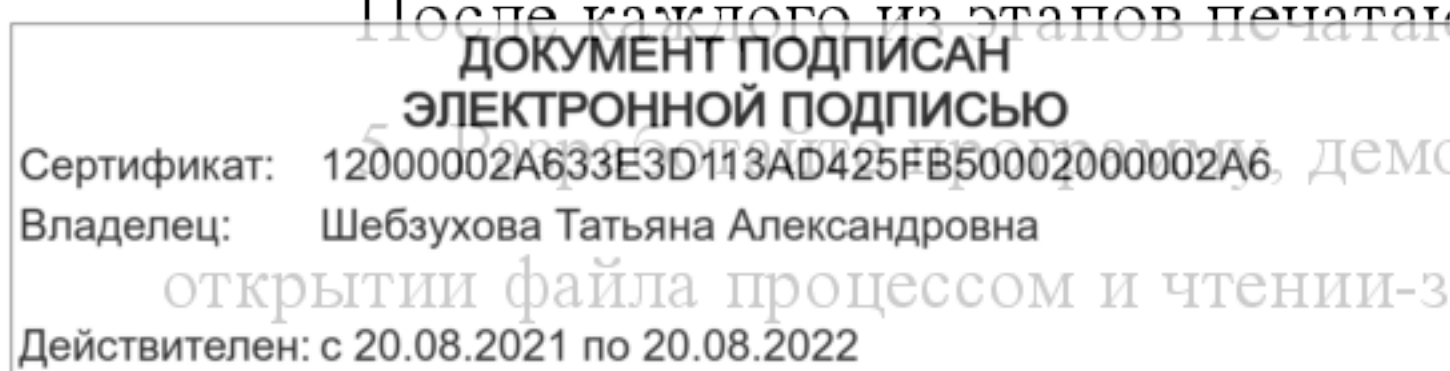
3. Пусть два процесса осуществляют доступ к одному и тому же файлу, но один из них читает файл, а другой пишет в него. Наступает момент, когда оба процесса обращаются к одному и тому же блоку диска. Пусть некоторая гипотетическая ОС использует ту же механику управления вводом-выводом, что и ОС UNIX, но не позволяет, как в ситуации, описанной выше, обращаться к одному блоку файла. Разработайте программу, которая демонстрирует "замораживание" перемещения указателя чтения-записи одного из процессов до тех пор, пока указатель второго процесса находится в этом блоке. Показать динамику создания всех таблиц, связанных с файлами и процессами, и изменение их содержимого.

После каждого из этапов печатаются таблицы файлов и открытых файлов обоими процессами.

4. Пусть N процессов осуществляют доступ к одному и тому же файлу на диске (но с разными режимами доступа). Разработать программу, демонстрирующую динамику формирования таблицы файлов и изменения ее элементов (при перемещении указателей чтения-записи, например). Например, сценарий программы может быть следующим:

- открытие файла процессом 0 для чтения;
- открытие файла процессом 1 для записи;
- открытие файла процессом 2 для добавления;
- чтение указанного числа байт файла процессом 0;
- запись указанного числа байт в файл процессом 1;
- добавление указанного числа байт в файл процессом 2.

После каждого из этапов печатаются таблицы файлов всех процессов.



демонстрирующую работу ОС UNIX при
открытии файла процессом и чтении-записи в него. При этом достаточно по-

казать только динамику создания таблиц, связанных с этим событием (таблица описателей файла, таблица файлов, таблица открытых файлов процесса). Например, сценарий программы может быть следующим:

- неявное открытие стандартного файла ввода;
- неявное открытие стандартного файла вывода;
- неявное открытие стандартного файла вывода ошибок;
- открытие первого пользовательского файла;
- открытие второго пользовательского файла;
- запись 20 байт в первый файл;
- чтение 15 байт из второго файла;
- запись 45 байт в первый файл.

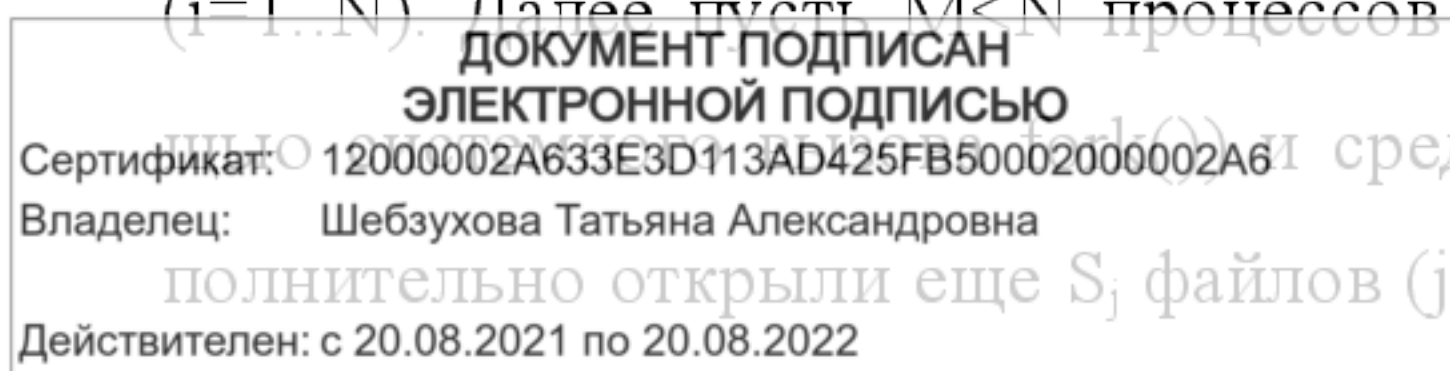
После каждого из этапов печатаются таблица описателей файлов, таблица файлов, таблица открытых файлов процессов.

6. Разработайте программу, демонстрирующую работу ОС UNIX при открытии файла процессом. При этом достаточно показать только динамику создания таблиц, связанных с этим событием (таблица описателей файла, таблица файлов, таблица открытых файлов процесса). Например, сценарий программы может быть следующим:

- неявное открытие стандартного файла ввода;
- неявное открытие стандартного файла вывода;
- неявное открытие стандартного файла вывода ошибок;
- открытие первого пользовательского файла;
- открытие второго пользовательского файла;
- открытие третьего пользовательского файла.

После каждого из этапов печатаются таблица описателей файлов, таблица файлов, таблица открытых файлов процессов.

7. Пусть каждый из N процессов осуществляет доступ к P_i файлам ($i=1..N$). Далее пусть $M < N$ процессов породили процессы-потомки (с помощью `fork()`) и среди этих потомков $K < M$ процессов дополнительно открыли еще S_j файлов ($j=1..K$). Разработать программу, демон-



стрирующую динамику формирования таблиц открытых файлов процессов. Например, сценарий программы может быть следующим:

- процесс 0 открывает два файла (общее число открытых файлов, включая стандартные файлы, равно пяти);
- процесс 1 открывает два файла (общее число открытых файлов, включая стандартные файлы, равно пяти);
- процесс 2 открывает два файла (общее число открытых файлов, включая стандартные файлы, равно пяти);
- процесс 0 порождает процесс 3, который наследует таблицу открытых файлов процесса 0;
- процесс 1 порождает процесс 4, который наследует таблицу открытых файлов процесса 1;
- процесс 4 дополнительно открыл ещё два файла.

После каждого из этапов печатаются таблицы открытых файлов процессов, участвующих в данном этапе.

8. Процесс создал новый файл и переназначил на него стандартный ввод. Разработайте программу, демонстрирующую динамику создания таблиц, связанных с этим событием (таблица описателей файла, таблица файлов, таблица открытых файлов процесса). Например, сценарий программы может быть следующим:

- неявное открытие стандартного файла ввода;
- неявное открытие стандартного файла вывода;
- неявное открытие стандартного файла вывода ошибок;
- чтение из стандартного файла ввода 5 байт;
- открытие пользовательского файла;
- закрытие стандартного файла ввода (моделирование `close(0)`);
- получение копии дескриптора пользовательского файла (моделирование `dup(fd)`, где `fd` – дескриптор пользовательского файла);

дескриптор пользовательского файла (моделирование `close(fd)`, где `fd` – дескриптор пользовательского файла);

чтение из "стандартного" файла ввода 10 байт.

После каждого из этапов печатаются таблица описателей файлов, таблица файлов, таблица открытых файлов процессов.

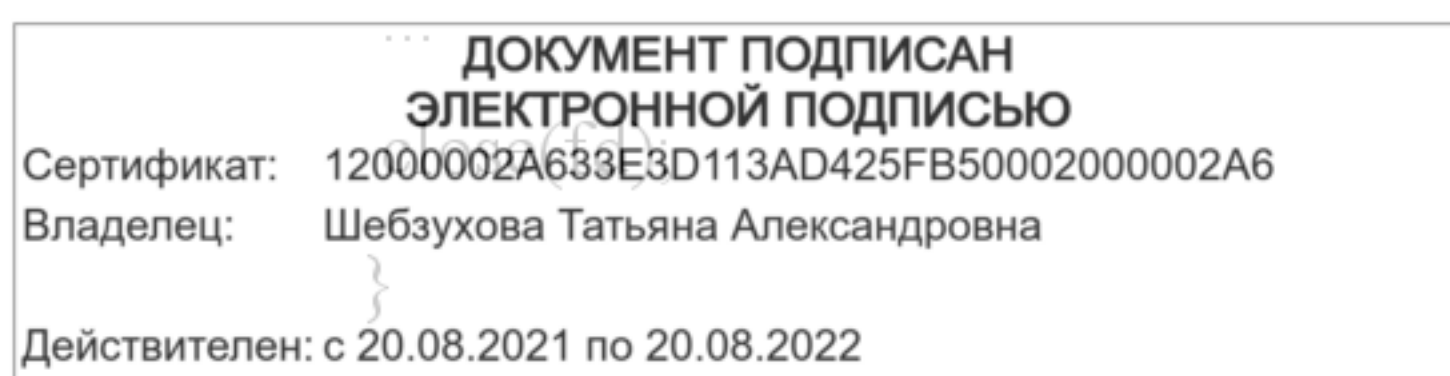
9. Пусть процесс, открывший N файлов, перед порождением процесса-потомка с помощью системного вызова `fork()` закрывает $K < N$ файлов. Процесс-потомок сразу после порождения закрывает $M < N - K$ файлов и через некоторое время завершается (в это время процесс-предок ожидает его завершения). Разработайте программу, демонстрирующую динамику изменения данных в системе управления вводом-выводом ОС UNIX (таблицы файлов и таблицы открытых файлов процессов). Например, сценарий программы может быть следующим:

- открытие процессом-предком стандартных файлов ввода-вывода и четырёх пользовательских файлов для чтения;
- закрытие процессом-предком двух пользовательских файлов;
- процесс-предок порождает процесс, который наследует таблицы файлов и открытых файлов процесса-предка;
- завершается процесс-потомок.

После каждого из этапов печатаются таблицы файлов и открытых файлов для обоих процессов.

10. Пусть процесс осуществляет действия в соответствии со следующим фрагментом программы:

```
main()
{
...
fd=creat(temporary, mode); /* открыть временный файл */
...
/* выполнение операций записи-чтения */
```



Разработайте программу, демонстрирующую динамику изменения данных системы управления вводом-выводом ОС UNIX (таблица описателей файлов, таблица файлов, таблица открытых файлов процесса).

Контрольные вопросы

1. Какова структура описателей файлов, таблицы файлов, таблицы открытых файлов процесса?
2. Какова цепочка соответствия дескриптора файла, открытого процессом, и файлом на диске?
3. Опишите функциональную структуру операции ввода-вывода (пулы, ассоциация их с драйверами, способы передачи информации и т.д.).
4. Каким образом осуществляется поддержка устройств ввода-вывода в ОС UNIX?
5. Какова структура таблиц открытых файлов, файлов и описателей файлов после открытия файла?
6. Какова структура таблиц открытых файлов, файлов и описателей файлов после закрытия файла?
7. Какова структура таблиц открытых файлов, файлов и описателей файлов после создания канала?
8. Какова структура таблиц открытых файлов, файлов и описателей файлов после создания нового процесса?

**ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ**

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по организации самостоятельной работы студентов
по дисциплине

«Управление информационными ресурсами обработки цифрового контента»
для направления подготовки **09.03.02 Информационные системы и
технологии**

направленность (профиль) **Информационные системы и технологии
обработки цифрового контента**

**Пятигорск
2022**

**ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ**

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

ВВЕДЕНИЕ

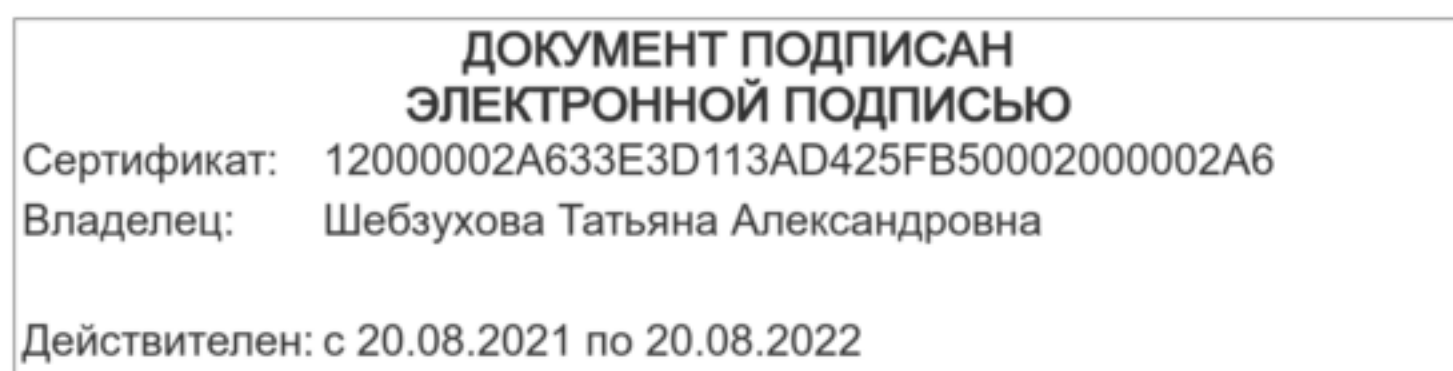
Управление информационными ресурсами обработки цифрового контента – дисциплина, изучающая основные принципы организации и методы управления аппаратными и программными средствами в вычислительных машинах (ВМ) и системах (ВС). В результате изучения курса студенты должны хорошо ориентироваться в принципах построения современных ВМ и ВС, архитектурных решениях, направленных на повышение производительности вычислительных машин, областях применения машин и систем с различной архитектурой и направлениях их развития.

Цели дисциплины:

- ознакомление студентов с основными принципами организации аппаратного обеспечения ЭВМ и систем, принципами работы периферийных устройств и их взаимодействия в составе системы.
- подготовка специалиста с современными знаниями об электронно-вычислительных машинах, высокой квалификации с широким теоретическим кругозором, способным осваивать новое в науке и технике.

Задачами дисциплины «Управление информационными ресурсами обработки цифрового контента» являются:

- 1) формирование комплекса знаний, умений и навыков, связанных с применением средств современной вычислительной техники, необходимых для правильного использования электронно-вычислительных машин и их модернизации;
- 2) формирование набора общепрофессиональных и профессиональных компетенций.



Содержание дисциплины

1. Основные понятия теории операционных систем: определение операционной системы, задачи и функции операционной системы, поколения операционных систем, классификация.

2. Понятие вычислительного процесса и ресурса: процесс и потоки; ресурсы и их классификация; состояние потока; реализация понятия последовательного процесса в теории операционных систем; описатель процесса; последовательное выполнение программ; механизмы последовательного выполнения.

3. Прерывания: механизм обработки прерываний; типы прерываний; маскирование прерываний; обработка прерываний в мультипрограммной системе.

4. Управление задачами и памятью: подсистема управления процессами и потоками; организация параллельно взаимодействующих процессов; управление памятью

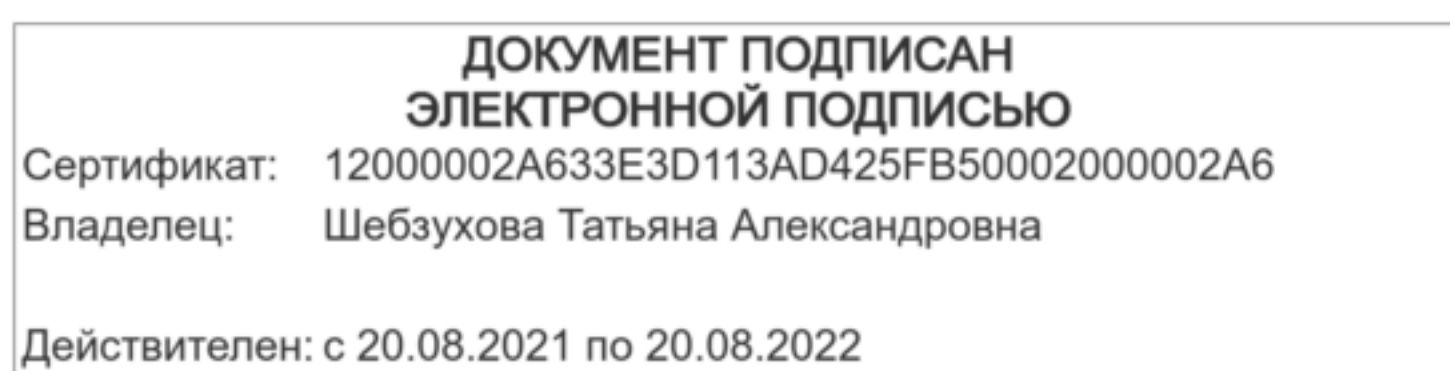
1. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лекционного и лабораторного занятия и изучение теоретического материала, приведенного в практической работе, а также выполнения индивидуального задания.

Подготовленный материал, практическое задание оформляются в конспекте.

Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине "Управление информационными ресурсами обработки цифрового контента" имеют целью:



- закрепить и углубить знания, полученные студентами на лекциях и в процессе выполнения лабораторных работ;

- сформировать практические навыки управления ресурсами ЭВМ.

Самостоятельные занятия проводятся в лабораториях, оборудованных современными средствами вычислительной техники и возможностью выхода в сеть Internet, в библиотеке.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Методические указания должны включать следующие разделы:

- цель работы;
- задание, которое должно быть выполнено студентом в результате проведения самостоятельной работы;
- варианты индивидуальных заданий;
- основные теоретические положения, необходимые для выполнения задания, они должны быть краткими и содержать ссылки на литературу, в которой эти положения изложены в объеме, достаточном для выполнения самостоятельной работы;
- этапы выполнения задания с указанием конкретных сроков выполнения каждого из этапов и всего задания в целом;
- требования к оформлению графической и текстовой части самостоятельной работы;
- пример выполнения одного из вариантов задания и оформления отчета;
- библиографический список использованных источников.

3. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

ДОКУМЕНТ ПОДПИСАН	
ЭЛЕКТРОННОЙ ПОДПИСЬЮ	
<i>3.1 Самостоятельное выполнение тем лекционного курса</i>	
Сертификат:	12000002A633E3D113AD425FB50002000002A6
Владелец:	Шебзухова Татьяна Александровна
<i>Раздел 1. Архитектура ЭВМ. Ресурсы ЭВМ</i>	
Действителен: с 20.08.2021 по 20.08.2022	

1. Методы кодирования информации.
2. Двоично-десятичный код.
3. Помехозащищенное кодирование.
4. Циклические коды.
5. Циклические избыточные коды.
6. Код Хемминга.
7. Циклические избыточные коды

Раздел 3. Управление локальными ресурсами

1. Системная плата, ее назначение, основные элементы и их взаимодействие в системе.
2. Системная магистраль.
3. Основные стандарты системных магистралей (шин).
4. Буферизация шин.
5. Управление системной магистралью.
6. Адаптеры внешних устройств (платы расширения).
7. Дисковая память.
8. Звуковые карты.
9. Мониторы.
10. Видеокарты.

3.2 Коллоквиум

Коллоквиум – это форма проверки и оценивания знаний учащихся в системе образования и представляет собой мини-экзамен, проводимый в середине семестра и имеющий целью уменьшить список тем, выносимых на дифференцированный зачет или экзамен.

Коллоквиум как одна из форм самостоятельной работы решает следующие задачи:

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ	
Сертификат:	12000002A633E3D113AD425FB50002000002A6
Владелец:	Шебзухова Татьяна Александровна
Действителен: с 20.08.2021 по 20.08.2022	

– проверка и контроль знаний, полученных студентами по изучаемой теме,

- расширение проблематики в рамках дополнительных вопросов по данной теме;
- углубление знаний при помощи дополнительных материалов при подготовке к занятию;
- студенты должны продемонстрировать умения работать с различными видами литературных источников;
- формирование навыков коллективного обсуждения (поддерживать диалог в микрогруппах, находить компромиссное решение и т.д.).

Коллоквиум может проводиться в двух формах – устной и письменной.

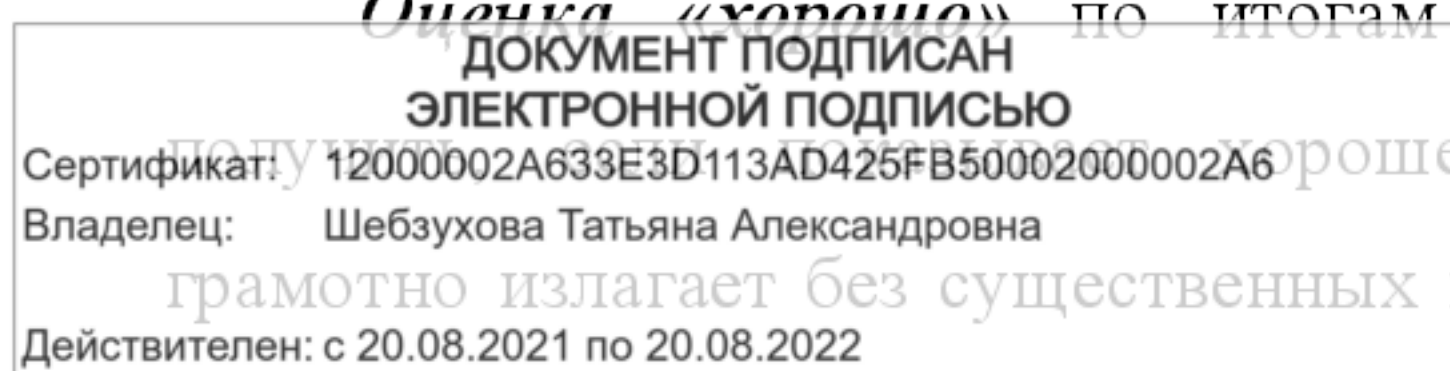
При проведении коллоквиума в устной форме ответы студентов оцениваются по традиционной шкале («неудовлетворительно» – «отлично») по билетам, которые могут содержать как теоретические вопросы, так и практические задания. На коллоквиум рекомендуется выносить лишь часть теоретического материала. Полученная по итогам его сдачи оценка влияет на итоговую по дисциплине.

Коллоквиум в письменной форме также проводится по билетам, в который могут быть включены теоретические вопросы, предполагающие короткий ответ, а также практические задания.

Критериями оценки знаний студентов на коллоквиуме являются:

Оценка «отлично» может быть выставлена, если студент демонстрирует глубокое и прочное усвоение программного материала, дает полные, последовательные грамотные и логически излагаемые ответы при видоизменении задания, свободно справляется с поставленными задачами и правильно применяет знания материала, четко обосновывает принятые решения, владеет разносторонними навыками и приемами выполнения практических работ.

Оценка «хорошо» по итогам сдачи коллоквиума студент может



хорошее знание программного материала, грамотно излагает без существенных неточностей в ответ на поставленный

вопрос, правильно применяет теоретические знания, владеет необходимыми навыками при выполнении практических задач.

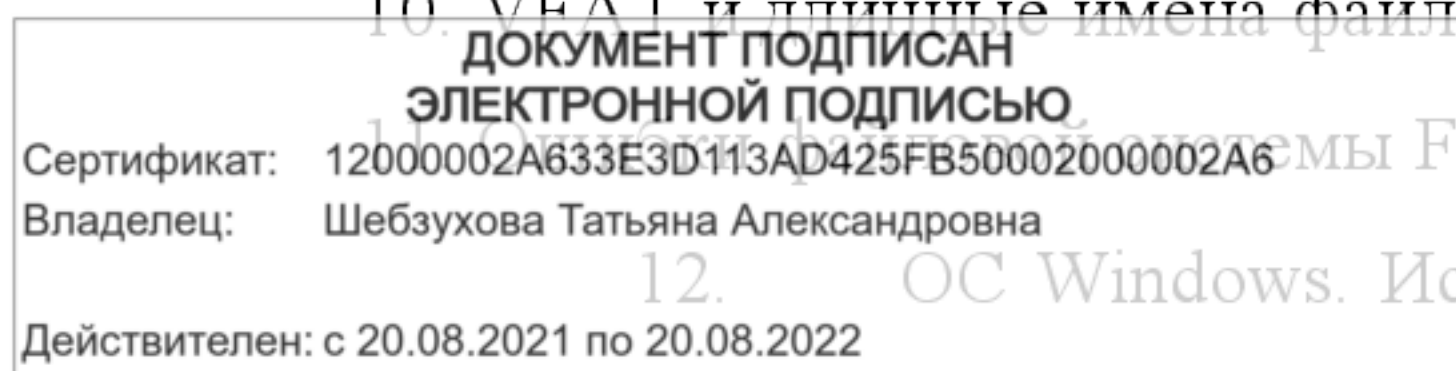
Оценка «удовлетворительно» может быть выставлена, если студент показал удовлетворительный уровень усвоения основного материала, допускает неточности при ответе на поставленные вопросы и в ответе присутствуют недостаточно правильные формулировки, нарушает последовательность в изложении программного материала, испытывает затруднения в выполнении практических заданий

Оценка «неудовлетворительно» по итогам сдачи коллоквиума выставляется, если студент демонстрирует полное не знание программного материала, допускает при ответе на вопрос и выполнении практических заданий грубые ошибки.

Коллоквиум по дисциплине «Управление информационными ресурсами обработки цифрового контента» включает в себя вопросы разделов «Управление ресурсами операционных систем» и «Управление локальными ресурсами» проводится в форме устного опроса.

Примерные вопросы к коллоквиуму

1. Файловая система. Понятие, виды, структура.
2. Загрузочный сектор.
3. Логическая структура жесткого диска.
4. Структура загрузочного сектора главного раздела.
5. Файловая система FAT.
6. Особенности и структура файловой системы FAT.
7. Виды FAT.
8. Корневой каталог.
9. Таблица размещения файлов.
10. VFAT и длинные имена файлов.
11. Особенности и структура файловой системы FAT.
12. ОС Windows. История, основные особенности, файловые системы, введение в безопасность операционной системы.



13. ОС Unix. История, основные особенности, файловые системы, введение в безопасность операционной системы.

4. ПЛАН-ГРАФИК ВЫПОЛНЕНИЯ СРС

№№	Форма самостоятельной работы	Срок ее сдачи
1	Индивидуальное собеседование по результатам самостоятельного изучения тем лекций	по итогам изучения соответствующего раздела лекционного курса
2	Коллоквиум	по итогам изучения разделов 2 и 3

5. ОРГАНИЗАЦИЯ КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

5.1. Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на практических занятиях и при подготовке к сдаче зачета. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Управление информационными ресурсами обработки цифрового контента», его способность использовать приближенные методы для решения научных и практических задач.

Контроль знаний студентов может осуществляться в следующих формах:

- текущий контроль знаний;
- итоговый контроль знаний.

Текущий контроль знаний студентов имеет целью:

- дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении;

Основными формами текущего контроля знаний студентов являются:

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ	
Сертификат:	12000002A633E3D113AD425FB50002000002A6
Владелец:	Шебзухова Татьяна Александровна
Действителен: с 20.08.2021 по 20.08.2022	

- письменный контрольный блиц-опрос;

- защита лабораторных работ;
- проверка конспектов лекций.

Устный контрольный опрос студентов проводится на лекциях и практических занятиях. По его результатам преподаватель оценивает качество подготовки студента к занятию.

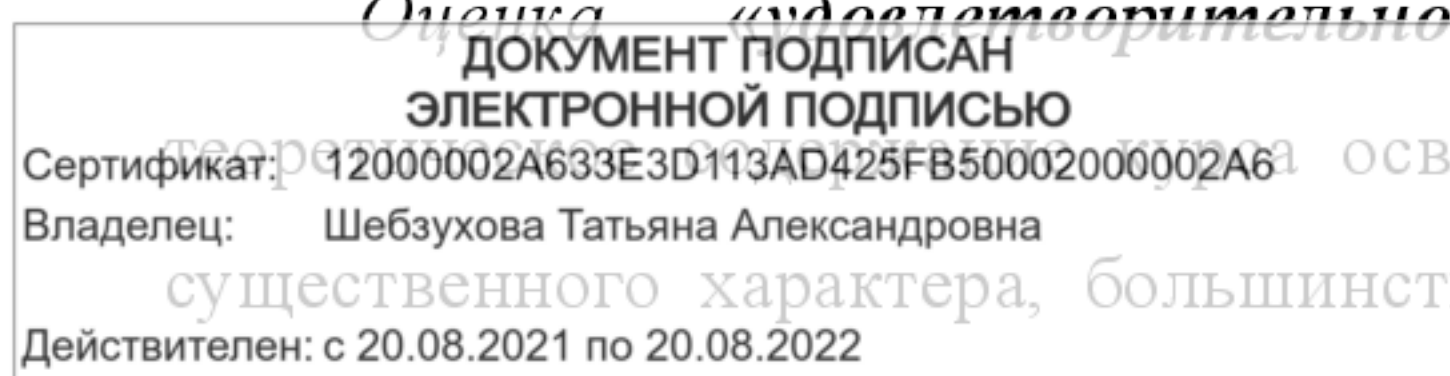
Письменный контрольный блиц-опрос студентов проводится в течение пяти минут на лабораторных занятиях путем письменного ответа их на пять вопросов, заданных преподавателем. Результаты его проведения отмечаются в журнале. На лабораторных занятиях знания и практические навыки студентов оцениваются по 5-балльной системе. Полученные оценки выставляются в журнале.

При этом для оценки сформированных у студентов компетенций используются следующие критерии:

Оценка «отлично» выставляется студенту, если теоретическое содержание курса освоено полностью, без пробелов; исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует в ответе дополнительный материал все предусмотренные программой задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному; анализирует полученные результаты; проявляет самостоятельность при выполнении заданий.

Оценка «хорошо» выставляется студенту, если теоретическое содержание курса освоено полностью, необходимые практические компетенции в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос.

Оценка «удовлетворительно» выставляется студенту, если теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, большинство предусмотренных программой



заданий выполнено, но в них имеются ошибки, при ответе на поставленный вопрос студент допускает неточности, недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении программного материала.

Оценка «неудовлетворительно» выставляется студенту, если он не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, необходимые практические компетенции не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, качество их выполнения оценено числом баллов, близким к минимальному.

Максимально возможный балл за весь текущий контроль устанавливается равным **55**. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

При проверке конспектов лекций дается анализ качества их ведения. Отмечаются допущенные ошибки, в рецензии преподавателя оценивается качество конспектирования учебного материала, даются рекомендации по улучшению качества конспектирования лекционного материала.

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ	
Сертификат:	12000002A633E3D113AD425FB50002000002A6
Владелец:	Шебзухова Татьяна Александровна
Действителен: с 20.08.2021 по 20.08.2022	

Рекомендации по подготовке к зачету

Подготовка к зачету начинается с начала изучения дисциплины. Для получения зачета необходимо посещать все лекции и лабораторные занятия, а так же предоставить отчеты о выполнении лабораторных работ.

6. РЕКОМЕНДАЦИИ ПО РАБОТЕ С ЛИТЕРАТУРОЙ И ИСТОЧНИКАМИ

6.1. Рекомендации студентам по организации работы с литературой и источниками

Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Управление информационными ресурсами обработки цифрового контента», которые прямо или косвенно относятся к изучаемой теме.

Изучая литературу и источники, студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные на ваш взгляд мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, студенту следует выписать их в виде цитат с полным указанием библиографических источников.

Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам.

Основой интенсификации обучения на платформе являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине "Управление информационными ресурсами обработки цифрового контента"

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022

Работа с учебно-иллюстрационными материалами имеет следующие этапы.

1. Изучение теоретических основ учебного материала в аудитории: изложение преподавателем изучаемого материала студентам с объяснением по опорному конспекту;

2. Самостоятельная работа: индивидуальная работа студентов по опорному конспекту; фронтальное закрепление по блокам опорного конспекта.

3. Первое повторение – воспроизведение содержания заданной темы опорного конспекта по памяти.

4. Устное проговаривание материала опорного конспекта – необходимый этап внешнеречевой деятельности при усвоении учебного материала.

5. Второе повторение – взаимопрос и взаимопомощь студентов друг другу.

Применение учебно-иллюстрационных материалов позволяет обобщить сложный по содержанию материал, активизировать мыслительную деятельность студентов.

Необходимо помнить, что главное для студента в самостоятельной работе с рекомендуемой литературой и источниками – это формирование своего индивидуального стиля, который может стать основой в будущей профессиональной деятельности.

6.2. Перечень рекомендуемой литературы

Основная литература

1. Н. В. Максимов, Партыка Т. Л., Попов И. И. Архитектура ЭВМ и вычислительных систем. – 3-е изд. – М. Форум – Инфра – М, 2010. – 512 с

2. Д. В. Игнатов Введение в операционные системы. Учебное пособие.

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат: 12000002A633E3D113AD425FB50002000002A6
Владелец: Шебзухова Татьяна Александровна
3. А. Ю. Молчанов Системное программное обеспечение. Учеб. Для
Действителен: с 20.08.2021 по 20.08.2022

ВУЗов. – 3-е изд. – СПб.: – 2010. – 400 с

Дополнительная литература

1. Таненбаум Э. Архитектура компьютера 5-е изд. – СПб.: Питер, 2007. – 844 с.
2. Мураховский В. И. Железо ПК. Новые возможности. – СПб.: Питер, 2005. – 592 с.
3. Таненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.

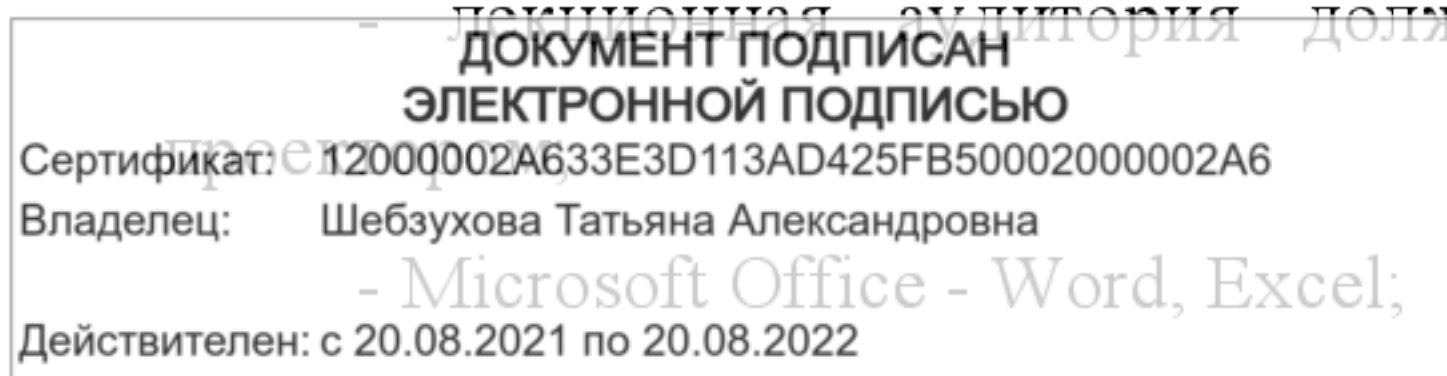
6.3. Internet-ресурсы:

- 1) <http://www.intuit.ru> – Национальный открытый университет «ИНТУИТ»;
- 2) <http://www.window.edu.ru> – Единое окно доступа к образовательным ресурсам.

6.4 Материально-техническое обеспечение дисциплины

К материально-техническому уровню освоения содержания дисциплины «Управление информационными ресурсами обработки цифрового контента» можно отнести:

- индивидуальное взаимодействие со студентами в процессе проведения лекционных и лабораторных занятий;
- использование на лекциях и лабораторных занятиях мультимедийного оборудования;
- включение в лабораторные работы индивидуального поиска, систематизации и анализа информации через Internet;
- авторские презентации к лекциям;
- лекционная аудитория должна быть оборудована мультимедиа



- ОС Windows;

- компьютерный класс с предустановленным программным обеспечением.

**ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ**

Сертификат: 12000002A633E3D113AD425FB50002000002A6

Владелец: Шебзухова Татьяна Александровна

Действителен: с 20.08.2021 по 20.08.2022