

Системное программное обеспечение

Задания к лабораторным работам

Содержание

Общие требования к программной реализации работ	2
Карта вариантов к лабораторным работам	2
Лабораторная работа №1 – файловые системы.....	3
Описание работы.....	3
Варианты заданий	3
Пример сеанса работы.....	4
Лабораторная работа №2 – интеграция компонентов низкого и высокого уровня	5
Описание работы.....	5
Варианты заданий	5
Лабораторная работа №3 – Удалённое взаимодействие	6
Описание работы.....	6
Варианты заданий	6
1. Чат.....	6
2. Доска обсуждений (Discussion-board).....	7
3. File exchange	8
4. Веб-сервер.....	8
5. Список задач (Task-list)	9
6. Library catalog	9
7. Remote FS browser	10
Отчеты	11

Общие требования к программной реализации работ

Для реализации необходимо использовать язык программирования Си, если в описании к лабораторной работе не сказано иное.

Исходный код каждой лабораторных работы размещать в отдельном репозитории на Gitlab факультета ПИИКТ (<https://gitlab.se.ifmo.ru/>). Ссылку на репозиторий добавить в отчет к работе.

Также необходимо следовать данным пунктам:

1. **«Нет» статичности.** Все структуры данных должны допускать создание множества их экземпляров, а функции – повторное использование.
2. **«Нет» «магическим» константам.** Все значения должны либо вычисляться из обрабатываемых программой данных, либо задаваться с помощью аргументов командной строки или конфигурационных файлов.
3. **«Нет» бесконечным циклам.** Все циклы должны иметь понятные условия выхода: не допускается использовать, например, `while (true)`, `for (; ;)` и т.д.
4. **«Нет» утечке ресурсов.** Все ресурсы, которые были использованы в программе и требуют освобождения (закрытия), должны корректно освобождаться (закрываться) независимо от возникновения ошибочных ситуаций или исключений. Например, открытый файл должен быть закрыт после того, как он перестал использоваться в программе; аллоцированная вручную память обязательно должна освобождаться.
5. **«Нет» неожиданным завершениям программы.** Все процессы, нити (threads) должны корректно завершаться в результате выполнения работы, а не прерываться функциями вида `Abort/Exit`.
6. **«Нет» побайтовому вводу-выводу.** Все данные должны обрабатываться частями (блоками) известного размера, с учетом целесообразного размера буфера.

Карта вариантов к лабораторным работам

Номер	Вариант задания				Номер	Вариант задания			
	ЛР1	ЛР2		ЛР3		ЛР1	ЛР2		ЛР3
1	1	1		1	22	6	4		1
2	2	2		2	23	7	5		2
3	3	3		3	24	8	6		3
4	4	4		4	25	1	7		4
5	5	5		5	26	2	8		5
6	6	6		6	27	3	9		6
7	7	7		7	28	4	1		7
8	8	8		1	29	5	2		1
9	1	9		2	30	6	3		2
10	2	1		3	31	7	4		3
11	3	2		4	32	8	5		4
12	4	3		5	33	1	6		5
13	5	4		6	34	2	7		6
14	6	5		7	35	3	8		7
15	7	6		1	36	4	9		1
16	8	7		2	37	5	1		2
17	1	8		3	38	6	2		3
18	2	9		4	39	7	3		4
19	3	1		5	40	8	4		5
20	4	2		6	41	1	5		6
21	5	3		7	42	2	6		7

Лабораторная работа №1 – файловые системы

Цель – изучение способов организации файловых систем.

Описание работы

Реализовать программу, которая может использоваться в двух режимах. Режимы задаются в виде аргументов командной строки и позволяют:

1. Выводить список дисков и разделов, подключенных к операционной системе.
2. Выполнять операции над файловой системой, представленной на заданном диске, разделе или в файле.

Запущенная во втором режиме программа должна выполнять следующие действия:

1. Проверять, поддерживается ли файловая система на заданном разделе или диске.
2. В случае, если файловая система поддерживается, программа переходит в диалоговый режим, ожидая ввода команд от пользователя. Команды задают операции над файловой системой:
 - a. вывод списка имен и атрибутов элементов указанной директории;
 - b. копирование файлов или директорий из исследуемой (заданной по варианту) файловой системы;
 - c. отображение названия «текущей» директории и переход в другую директорию.

Программа должна состоять из двух модулей. Первый модуль реализует функции для работы с файловой системой, а второй – взаимодействие с пользователем.

Варианты заданий

Вариант	ФС	Источники документации
1	FAT32	https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc776720(v=ws.10) https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system
2	NTFS	https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc758691(v=ws.10)?redirectedfrom=MSDN Книга: Windows NT File System Internals: A Developer's Guide Paperback – September 11, 1997
3	Ext3	https://students.mimuw.edu.pl/SO/Linux-doc/UnderstLinuxKernel2e-ch17.pdf Книга: Understanding the Linux Kernel Daniel P. Bovet and Marco Cesati
4	Btrfs	https://btrfs.wiki.kernel.org/index.php/Btrfs_design https://btrfs.wiki.kernel.org/index.php/On-disk_Format https://btrfs.wiki.kernel.org/index.php/Data_Structures
5	HFS Plus	https://developer.apple.com/library/archive/technotes/tn/tn1150.html https://github.com/libyal/libfshfs/blob/main/documentation/Hierarchical%20File%20System%20(HFS).asciidoc
6	Reiser	http://jcoppens.com/univ/data/pdf/fs/reiserfs.pdf https://reiser4.wiki.kernel.org/index.php/Main_Page
7	Xfs	http://ftp.ntu.edu.tw/linux/utlis/fs/xfs/docs/xfs_filesystem_structure.pdf https://xfs.org/docs/xfsdocs-xml-dev/XFS_Filesystem_Structure/tmp/en-US/html/index.html
8	UFS/FFS	https://flylib.com/books/en/2.849.1.110/1/ http://callibra.com.br/Violino/Carrier,%20Brian%20-%20File%20System%20Analysis.pdf

Пример сеанса работы

```
user@host:~$ fsbrowse -l
/dev/sda
/dev/sda1
/dev/sda2
user@host:~$ fsbrowse -li
/dev/sda          100 GB
/dev/sda1         3.7 GB
/dev/sda2         96.3 GB   ext3
user@host:~$ fsbrowse /dev/sda1
Unknown filesystem.
user@host:~$ fsbrowse /dev/sda2
# pwd
/
# ls
test1/
otherDirInFsRoot/
somefile
# cd /test1/test2
No such directory
# cd /test1
# ls
subdir1/
subdir2/
file1
file2
file3
# ls subdir2
testfile4
testfile5
testfile6
# pwd
/test1
# cp ./test2 /tmp/
Extracting /test1/test2/testfile4 to /tmp/test2/testfile4
Extracting /test1/test2/testfile5 to /tmp/test2/testfile5
Extracting /test1/test2/testfile6 to /tmp/test2/testfile6
# exit
user@host:~$ ls /tmp/test2
testfile4
testfile5
testfile6
user@host:~$
```

Лабораторная работа №2 – интеграция компонентов низкого и высокого уровня

Цель – изучение способов организации программных интерфейсов между средами высокого и низкого уровней программной архитектуры.

Описание работы

Построить разделяемую библиотеку (shared library) с функциональностью для работы с файловой системой, реализованную в лабораторной работе №1.

На языке высокого уровня реализовать консольное или графическое приложение, функциональность которого аналогична программе из лабораторной работы №1, для операций с файловой системой использовать полученную библиотеку, написанную на Си. При необходимости реализовать дополнительную библиотеку, обеспечивающую вызов функций из программы на языке высокого уровня.

Варианты заданий

Варианты языков программирования высокого уровня:

1. C# (или другой .NET-совместимый язык по согласованию с преподавателем).
2. Java (Scala/Kotlin или другой JVM-совместимый язык по согласованию с преподавателем).
3. Python.
4. JavaScript (или Typescript, другой язык по согласованию с преподавателем) + Node.js.
5. PHP.
6. Perl.
7. Ruby.
8. Erlang.
9. Lua.

Лабораторная работа №3 – Удалённое взаимодействие

Цель – изучение способов взаимодействия между сетевыми службами низкого уровня в асинхронном режиме.

Описание работы

Разработать клиент-серверное приложение. Для организации взаимодействия по сети, поддержки множества соединений использовать программные интерфейсы (API) операционной системы.

Сервер и клиенты взаимодействуют по протоколу, реализованному на базе сокетов (использовать TCP, если в варианте задания не указано обратное). Сервер должен поддерживать условно неограниченное количество клиентов. На всех этапах взаимодействия клиента и сервера должна быть предусмотрена обработка данных независимо от их размера.

Порядок выполнения:

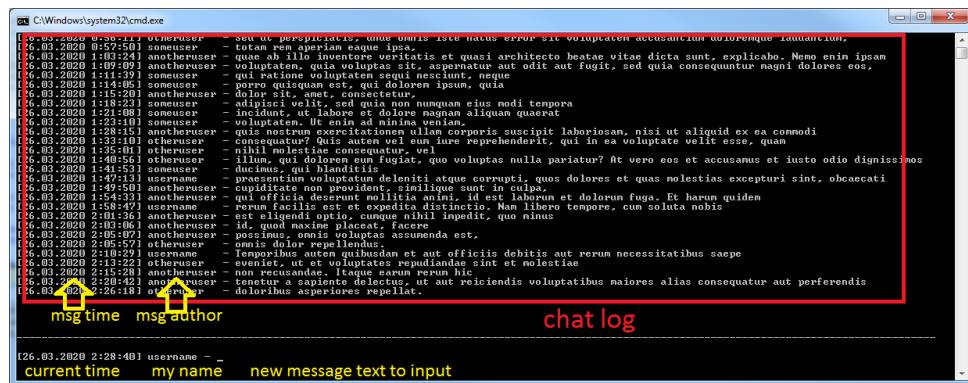
1. Выполнить анализ предметной области, которая задается вариантом к лабораторной работе. Результатом анализа должен быть набор сущностей, которые будут в качестве элементов данных (типов, структур, операций) и/или составных частей программной архитектуры (компонентов, модулей) при реализации программы.
2. Составить диаграмму, на которой схематически будут показаны результаты анализа: сущности, их атрибуты и взаимосвязи.
3. Составить план постепенного выполнения задания: какие части функциональности, в каком порядке предполагается реализовывать, в каком порядке и как проверять их работоспособность.
4. Загрузить все полученные артефакты в отдельную директорию «docs» репозитория, в корневую директорию положить readme.md с номером варианта и кратким описанием.
5. Продемонстрировать составленные диаграмму и план преподавателю. Для этого достаточно просто отправить преподавателю ссылку на репозиторий.
6. После проверки и получения рекомендаций приступить к реализации программы, создавая на каждый этап выполнения отдельную ветку в репозитории.
7. По завершении каждого этапа создать pull-request, включив преподавателя в число reviewer-ов.
8. Если pull-request отклоняется преподавателем, выполнить необходимые правки и обновить его, запросив повторное ревью.
9. Когда pull-request одобрен, «слить» его с основной веткой кода, после чего создать новую ветку для работы над следующим этапом.

Варианты заданий

1. Чат

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. В режиме сервера линейно отображается журнал всех входящих и исходящих сообщений, завершение программы-сервера выполняется по нажатию ключевой клавиши (например, Q).

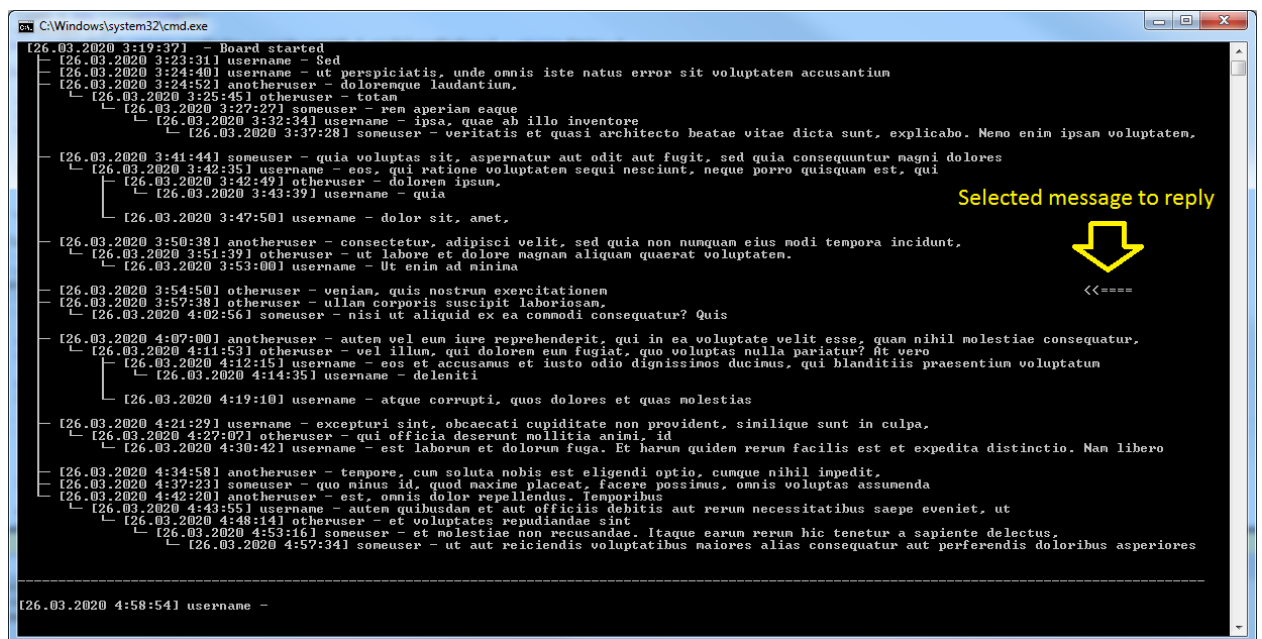
При запуске в режиме клиента программе в качестве аргументов командной строки также передается имя пользователя и адрес сервера. Необходимо предусмотреть возможность отправки «приватного» сообщения, которое увидит только адресат, которому оно предназначено. При подключении отображать последние 20 сообщений, предусмотреть возможность просмотра истории сообщений (не сохраняя её при этом на стороне клиентского приложения).



Новые сообщения выводятся в конце списка с автопрокруткой по мере появления новых сообщений. Клавишами «стрелка вверх/вниз», Page Up/Page Down выполняется прокрутка списка сообщений, отменяющая автопрокрутку при появлении новых сообщений. Клавишей End автопрокрутка возобновляется. Собственное сообщение вводится в отдельной строке в нижней части окна, не блокируя историю сообщений.

2. Доска обсуждений (Discussion-board)

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. В режиме сервера линейно отображается журнал всех входящих и исходящих сообщений, завершение программы-сервера выполняется по нажатию ключевой клавиши (например, Q).



При запуске в режиме клиента программе в качестве аргументов командной строки также передается имя пользователя и адрес сервера. Дискуссионная доска представляет собой дерево сообщений, узлы которого можно разворачивать или сворачивать для просмотра ответов на интересующее сообщение. Для отправки своего сообщения в дереве выбирается существующее, под которым появится отправляемый ответ. Предусмотреть прокрутку дерева, если оно не помещается на экране, а также индикацию появления новых ответов, в том числе в неразвёрнутых ветках. Собственное сообщение вводится в отдельной строке в нижней части окна, не блокируя работу дерева.

3. File exchange

Один совмещённый режим работы. При запуске аргументом командной строки указывается путь к директории для работы (обслуживания). В момент запуска сервер рекурсивно сканирует указанную директорию и вычисляет хэши для всех файлов. Эти файлы программа будет давать возможность скачивать другим подключающимся к ней экземплярам. Параллельно начинается приём входящих соединений и ожидание ввода команд из консоли.

На экран интерактивно выводится список скачиваемых и отдаваемых файлов с прогрессом в процентах и байтах, область ввода команды. Обработывается три команды:

1. Вывод триплета «имя файла – размер - хэш» для относительного пути файла из рабочей директории.
2. Ввод триплета «имя файла - размер - хэш» для скачивания.
3. Завершение программы.

Downloading		Uploading	
filename	12/22MB 55%[xxxx.....]	filename	12/22MB 55%[xxxx.....]
....		status from downloader side	
Actions/events log			
[datetime] Downloading filename started			
[datetime] Downloading otherfile finished			
[datetime] Uploading filename to somehostname started			
autoscroll here			
> _		command input field	

При вводе команды на скачивание с помощью широковещательной UDP-рассылки обнаруживаются другие экземпляры программы в сети, ответным пакетом от них получается порт для подключения по TCP. После установки соединения происходит запрос файла по триплету, в случае его обнаружения начинается скачивание. Предусмотреть одновременное скачивание разных фрагментов одного искомого файла из нескольких источников.

4. Веб-сервер

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки.

При запуске в режиме сервера аргументом командной строки указывается путь к директории для работы (обслуживания). Завершение программы-сервера осуществляется по нажатию ключевой клавиши (например, Q). В любой директории, начиная с рабочей, может быть расположен файл, содержащий управляющие инструкции. По умолчанию сервер возвращает запрашиваемые по HTTP файлы относительно рабочей директории. При этом управляющими инструкциями задаётся следующая дополнительная информация:

- отображение расширения файла на имя MIME-типа, который отдаётся в заголовках;
- CGI-обработчик или интерпретатор для файлов (скриптов) по расширению в соответствии;
- запрет отдавать файлы с фильтром по регулярному выражению.

В соответствии с HTTP/1.1 через одно соединение может быть передан ряд запросов. Помимо GET, приём тела при обработке PUT/POST запросов на скрипт. В консоль при этом выводится информация о прогрессе отдаваемых ресурсов аналогично клиенту при получении.

```
Resolving ya.ru (ya.ru)... 87.250.250.242, 2a02:6b8::2:242
Resolving releases.ubuntu.com (releases.ubuntu.com)... 91.189.88.248, 91.189.91.123, 91.189.91.124, ...
index.html          100%[=====>] 20.65K --.-KB/s   in 0.02s
ubuntu-20.04-desktop-amd64.iso  7%[====>] 198.30M 3.92MB/s   eta 10m 24s
```




При запуске в режиме клиента через аргументы командной строки задаётся URL ресурсов для получения и опционально имена для сохраняемого файлов. Предусмотреть опциональную возможность скачивания связанных ресурсов (если запрашиваемый ресурс – html-страница) – с их сохранением в директории рядом. В зависимости от аргументов командной строки, выполнять скачивание ресурсов параллельно.

5. Список задач (Task-list)

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. Завершение программы-сервера происходит по нажатию ключевой клавиши (например, Q).

При запуске в режиме клиента через аргументы командной строки задаётся имя пользователя и адрес сервера.

Каждый подключающийся пользователь имеет свой, отдельный от других пользователей, набор списков задач. Каждый список задач описывается названием и состоит из задач, каждая задача представлена датой создания, кратким заголовком, описанием и опциональным планируемым временем завершения. Пользователь может создавать/уничтожать/редактировать листы задач и задачи в них. Повторные подключения того же пользователя должны так же отображать актуальное состояние его списков задач. Переключение между областями ввода по Tab, применение введенного значения по Enter.

list 1	list2 tasks:	task1 details
> list 2 list 3  selected tasklist	> task 1 task 2 task 3 task 4  selected task	created Sed ut perspiciatis, unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem deadline
> Sed ut perspiciatis, unde_omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem  current field value edit area		

6. Library catalog

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. Завершение программы-сервера происходит по нажатию ключевой клавиши (например, Q).

При запуске в режиме клиента чрез аргументы командной строки задаётся имя пользователя и адрес сервера.

The screenshot shows a terminal window with a book management application. The interface is divided into two main sections. The left section contains a list of books: book1, book2, book4, book5, and book6. A yellow arrow points to book1, labeled 'selected book'. Below the list is a 'search filter field'. The right section displays the details for the selected book (book1). It includes fields for 'Title: Lorem ipsum dolor sit amet', 'Authors: a, b, c', 'Annotation: Sed ut perspiciatis, unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam eaque ipsa, quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt, explicabo.', 'Tags: x, y, z', and 'Available: 5/10'. A yellow circle highlights the 'current book info fields'. At the top of the right section, there are three buttons: '[Get book]', '[Return book]', and '[Register new book]', with a yellow arrow pointing to them labeled 'action buttons'. At the bottom of the terminal, there is a filter command: 'filter by [x] title, [] author, [] annotation, [x] tags', with a yellow circle around it labeled 'filter mode checkboxes'.

Каждая книга описывается карточкой, содержащей сведения о названии, авторе, дате издания, аннотации, и наборе тэгов (таких как жанры и т.п.), количество приписанных к каталогу и количество в наличии. Каждый подключающийся пользователь может помещать книги в каталог, брать на прочтение и возвращать. Предусмотреть поиск сведений о книгах по различным критериям. Если описание книги редактируется или изменяется статус наличия, пользователям должна своевременно показываться актуальная информация по мере её изменения. Визуально любые операции редактирования осуществляются «in-place», то есть по месту визуализации данных поля.

7. Remote FS browser

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. При запуске в режиме сервера аргументом командной строки указывается путь к директории для работы. Завершение программы-сервера происходит по нажатию ключевой клавиши (например, Q).

При запуске в режиме клиента через аргументы командной строки задаётся адрес сервера.

Клиент отображает интерактивный список файлов и директорий в текущем расположении внутри рабочей директории сервера. Предусмотреть переход между директориями внутри поддерева, просмотр текстовых файлов, скачивание и загрузку произвольных файлов. При работе нескольких пользователей одновременно внутри одной директории отображать актуальное состояние.

C:\Windows>			
symbols	<Folder>	22.04.16	
system	<Folder>	14.07.09	
System32	<Folder>	13.10.18	
SysWOW64	<Folder>	02.03.17	
TAPI	<Folder>	14.07.09	
Tasks	<Folder>	01.10.19	
Temp	<Folder>	25.05.20	
tracing	<Folder>	14.07.09	
twain_32	<Folder>	14.07.09	
Vss	<Folder>	14.07.09	
Web	<Folder>	14.07.09	
winsxs	<Folder>	19.04.20	
bfsvc	exe	71,168	21.11.10
bootstat	dat	67,584	23.05.20
ctfile	rfc	78	30.03.20
cthdaeng	reg	4,850	02.04.12
cthdaloc	reg	4,850	02.04.12
directx	log	62,605	08.11.18
dteinstall	log	2,790	22.04.16
epplauncher	mif	1,945	14.05.16
explorer	exe	2,872,320	21.11.10

Про отчеты

К концу семестра по каждой из работ должен быть представлен отчет, содержащие следующие части:

1. Цели – описание цели задания (см. текст задания)
2. Задачи – путь достижения цели, что именно нужно было сделать для выполнения задания, входная информация для выполнения
3. Описание работы – внешнее описание созданной программы, состава модулей, способов её использования, примеры входной и выходной информации
4. Аспекты реализации – внутреннее описание созданной программы, особенности алгоритмов, примеры кода
5. Результаты – что было сделано для выполнения задач кратко по пунктам
6. Выводы – что было достигнуто в отношении цели задания.

Отчет в pdf должен быть загружен в репозиторий с артефактами по лабораторной работе.