Министерство науки и высшего образования Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления» Кафедра «Компьютерные системы и сети»

Утверждаю

Зав.каф. ИУ6

\_\_\_\_\_\_\_\_\_\_Пролетарский А.В.

«\_\_\_»\_\_\_\_\_\_\_ 2020 г.

Большие вычислительные машины (мейнфрейм):

архитектура и микроархитектура zSeries,

операционные системы

и

основы программирования

Методические указания

по выполнению лабораторных работ

по курсу БЭВМ

Издание третье, исправленное

2020 год

Москва

1

Содержание:

[Работа в среде БЭВМ с использованием эмулятора терминала Vista TN3270](#bookmark=id.1fob9te) [3](#bookmark=id.1fob9te)

[1 Подключение к серверу z/OS](#bookmark=id.3znysh7) [4](#bookmark=id.3znysh7)

[2 Вход в операционную систему z/OS](#bookmark=id.2et92p0) [5](#bookmark=id.2et92p0)

[3 Создание наборов данных (Data Set)](#bookmark=id.1t3h5sf) [8](#bookmark=id.1t3h5sf)

[4 Выход из операционной системы z/OS (z/OS Logoff)](#bookmark=id.35nkun2) [17](#bookmark=id.35nkun2)

[5 Контрольные вопросы](#bookmark=id.44sinio) [18](#bookmark=id.44sinio)

[Редактирование программ в среде Mainframe с использованием редактора ISPF](#bookmark=id.2jxsxqh) [20](#bookmark=id.2jxsxqh)

[1. Вход в операционную систему z/OS](#bookmark=id.30j0zll) [21](#bookmark=id.30j0zll)

[2. Подготовка к разработке программы](#bookmark=id.30j0zll) [21](#bookmark=id.30j0zll)

[3. Редактирование данных в режиме ISPF Editor (кодирование на языке С)](#bookmark=id.1fob9te) [22](#bookmark=id.1fob9te)

4. Создание JCL-скрипта и его выполнение 24



[5. Выход из операционной системы z/OS (z/OS Logoff)](#bookmark=id.1fob9te) [26](#bookmark=id.1fob9te)

[6. Контрольные вопросы](#bookmark=id.qsh70q) [28](#bookmark=id.qsh70q)

Создание клиент-серверных приложений на языке EGL, серверная часть программы 29

1. Теоретическая часть 30



2. Запуск RBDe, создание и настройка проекта 33

3. Создание основных модулей 40

4. Тестирование и отладка программы 51

[5. Контрольные вопросы](#bookmark=id.qsh70q) [55](#bookmark=id.qsh70q)

Создание клиент-серверных приложений на языке EGL, клиентская часть программы 56

1. Теоретическая часть 57

2. Запуск и начальная настройка RDBe 58



3. Создание и настройка проекта 58

4. Создание веб страниц 63

5. Редактирование модулей EGL JSFHandler 76

6. Тестирование и отладка программы 81

[7. Контрольные вопросы](#bookmark=id.qsh70q) [88](#bookmark=id.qsh70q)

Создание клиент-серверных приложений на языке EGL, генерация Java кода и запуск программы 89

1. Теоретическая часть 90

2. Запуск RDBe 90



3. Генерация Java кода для клиентской части программы 91

4. Генерация Java кода для серверной части программы 95

5. Запуск клиент-серверного приложения 100

[6. Контрольные вопросы](#bookmark=id.qsh70q) [104](#bookmark=id.qsh70q)

Системные сервисы z/OS UNIX 105

1. Теоретическая часть 106



Иерархическая файловая система HFS. 108

Скриптовый язык awk 111

Режимы доступа пользователей к z/OS UNIX 112

2. Практическая часть 112



2.1. Запуск оболочки shell системных сервисов UNIX 112



[2.2 Работа с файлами и каталогами](#bookmark=id.30j0zll) [114](#bookmark=id.30j0zll)



2.3. Изучение основных команд оболочки shell 118

2.4. Настройка оболочки shell 120

2.5. Написание shell скриптов 121

2.6. Создание awk программ 122

2.7. Создание С программ 125

3. Требования к отчету 125

[4. Контрольные вопросы](#bookmark=id.qsh70q) [126](#bookmark=id.qsh70q)

5. Список литературы 127

2

Министерство науки и высшего образования Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления» Кафедра «Компьютерные системы и сети»

Смирнова Е.В., Егоров М.А., Сергеев В. С.

Лабораторная работа №1

**Работа в среде БЭВМ с использованием эмулятора терминала Vista TN3270**

Методические указания по выполнению лабораторной работы

по курсу «Архитектура больших ЭВМ»

2020 год,

Москва

3

Лабораторная работа №1

Работа в среде БЭВМ с использованием эмулятора терминала Vista TN3270

**Цели лабораторной работы:**

1. Освоение способов подключения к виртуальной среде большой вычислительной машины Mainframe и работа в ее операционной системе z/OS.
2. Создание контейнеров данных (Data Set) для хранения файлов (Member).

**Выполнение лабораторной работы**

***1 Подключение к внутренней сети компании IBM (vpn)***

https://vpn.ibmcc.ru - адрес подключения в приложении VPN CISCO AnyConnect

Для подсоединения к сети IBM через VPN - Cisco AnyConnect нужно будет установить приложение на компьютер, скачать его можно здесь: <https://software.cisco.com/download/home/286281283/type/282364313/release/4.7.04056?i=!pp>

Интерфейс подключения показан на рисунке 1.

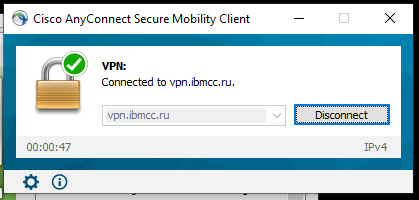


Рисунок 1

Логин и пароль для подключения студент получает у преподавателя.

***2 Подключение к серверу z/OS***

Операционная система z/OS поддерживает множество способов обращения к ней. Для того чтобы получить доступ к серверу z/OS с вашего компьютера, вам необходимо установить компонент доступа, так называемый Клиент. Примеры таких компонентов показаны на рисунке 2.

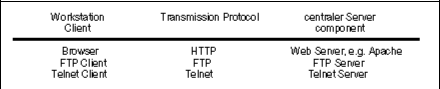


Рисунок 2 Компоненты доступа к z/OS.



* частности, вы можете использовать протокол IBM 3270, который обеспечивает связь с сервером коммуникаций z/OS. Этот протокол использует протокол TELNET, как носитель, и осуществляет доступ через TCP/IP порт 23. Клиент 3270 называют также **эмулятором терминала Mainframe** (рисунок 3).

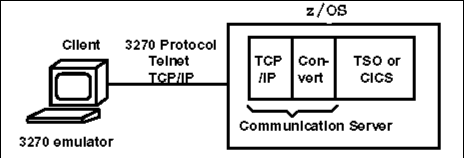


Рисунок 3 –

Клиент 3270 и сервер z/OS.



Эмулятор терминала мейнфрейма TN3270 вы можете скачать отсюда: <https://www.mochasoft.dk/tn3270.htm> (предоставляется 30 дней бесплатного использования). Окно запуска эмулятора терминала TN3279 дано на рисунке 4.

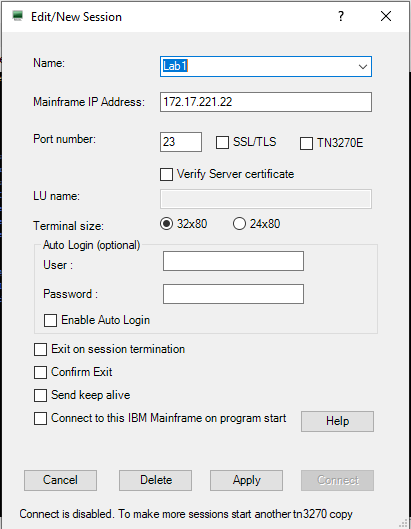


Рисунок 4 – Окно интерфейса эмулятора терминала – настройка сессии.

Установите и запустите эмулятор терминала, указав адрес сервера

**IP - 172.17.221.22 и Port 23.**

***2 Вход в операционную систему z/OS***

При удачном соединении на экране появится окно приветствия операционной системы z/OS (рисунок 5) и машина будет ожидать от Вас ввода Вашего логина (Для работы в операционной системе z/OS Вы должны получить **регистрационный** **номер** (UserID) и **пароль** (Password) у преподавателя).

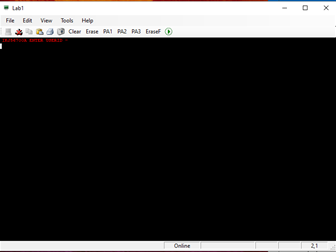


Рисунок 5 – Окно приветствия системы z/OS

.



.

На этом месте ВНИМАНИЕ!

ЧИТАЙТЕ ВНИМАТЕЛЬНО ВСЕ СООБЩЕНИЯ ОТ МЕЙНФРЕЙМА!

После ввода Вашего логина появится окно (рисунок 6). После ввода UserID перед вами появится приветственный экран **TSO/E** (рисунок 6). Введите пароль Password и нажмите еще раз **Enter,** и вы попадете в приветственный экран системы TSO ( рисунок 7).

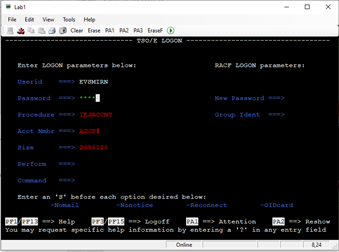


Рисунок 6 – Экран входа в операционную систему TSO/E

.



При первом запуске система попросит Вас сменить пароль на новый. Введите новый пароль поле **New Password** (не больше 7 символов!!!) и нажмите **Enter**. Подтвердите новый пароль, напечатав его в поле **New Password** еще раз, и нажмите **Enter**.

**Символ “\*\*\*”** говорит о том, что система готова вывести следующее сообщение. Каждый раз, когда Вы видите на экране символ “\*\*\*”, Вы можете не торопиться и прочитать сообщения системы, по нажатию клавиши Enter система выдаст продолжение информационного текста.

**Примечание: Это - важная информация!** Если вы будете сдавать тест на получение Сертификате сисадмина мейнфрейма - знайте, это ответ на один из вопросов теста.

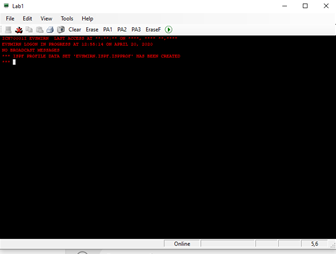


Рисунок 6

Нажмите еще раз **Enter,** и вы попадете в окно интерфейса **ISPF** (рисунок 8). Если не попали, нажмите F3 и введите слово ISPF.

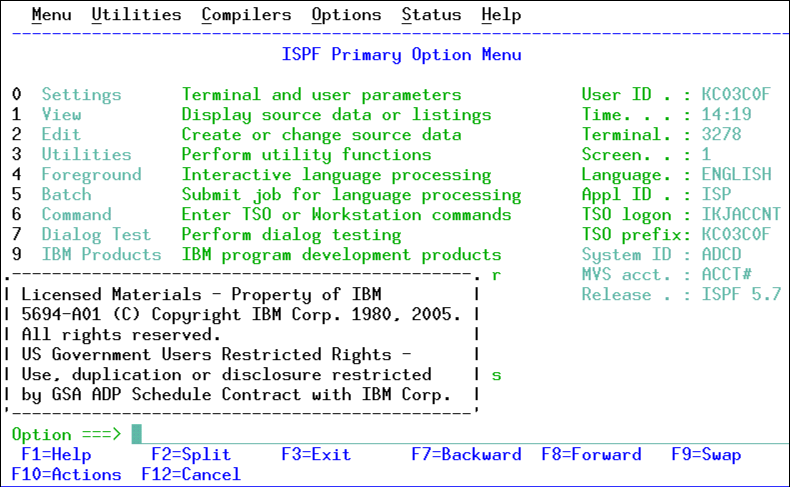


Рисунок 8 – Начальное окно интерфейса ISPF.



Подсистема **ISPF** построена как иерархическое меню, позволяющее вызывать различные функции **TSO**, вводя команды в командной строке. Операционная система **z/OS** управляет многими подсистемами. Иерархическая взаимосвязь подсистем и функций, которые вы будете осваивать, показана на рисунке 9. Для того, чтобы перейти к выбранному пункту меню, введите его номер в командной строке. Если вам нужно перейти сразу на несколько уровней меню вниз, вы можете ввести номера пунктов, разделенные точкой. Так, введя «3.2» в главном меню ISPF (рисунок 8), вы попадете в утилиту **Data Set** из меню **Utilites**.

Чтобы вернуться назад из подменю или утилиты, нажмите **F3.**

**Примечание:** Это - важная информация! Если вы будете сдавать тест на получение Сертификате сисадмина мейнфрейма - знайте, это ответ на один из вопросов теста.

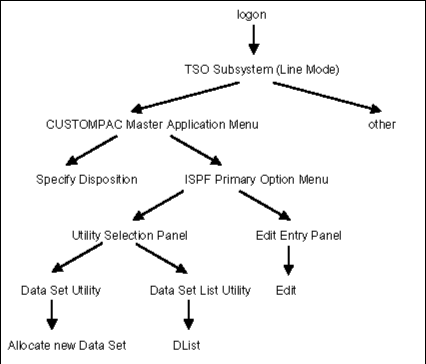


Рисунок 9 – Иерархическая структура подсистем и функций ISPF.

***3 Создание наборов данных (Data Set)***

Теперь создадим собственный набор данных.

**Набор данных** (data set) - именованная совокупность связанных элементов данных, размещаемых во внешней памяти или иных устройствах. Операционная система z/OS поддерживает работу с наборами данных различной логической организации: последовательными, прямого доступа, библиотечными (PDS и PDSE) некоторыми другими. Мы будем использовать библиотечный тип набора данных, так как он наиболее похож на «папку» OC Windows.

Библиотечные наборы данных (Partitioned Data Set, PDS), - это совокупность разделов (members), снабженная оглавлением (directory), в котором содержится информация об именах разделов и их размещении в памяти.

Структура набора данных распределенного типа (**Partitioned Data Set**) показана на рисунке 10. Такие наборы данных обычно используются для хранения относительно небольших по объему "блоков" информации: исходных текстов программ, процедур и заданий, объектных модулей, текстовых документов, таблиц и т.п.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Directory | Member 1 | Member 2 | Member 3 | Member 4 | ... |



Рисунок 10 – Структура набора данных распределенного типа (Partitioned Data Set).

Для создания набора данных из основного окна интерфейса **ISPF** (рисунок 8) войдем в меню **Utility Selection Panel**, нажав «**3**» в командной строке (рисунок 11).

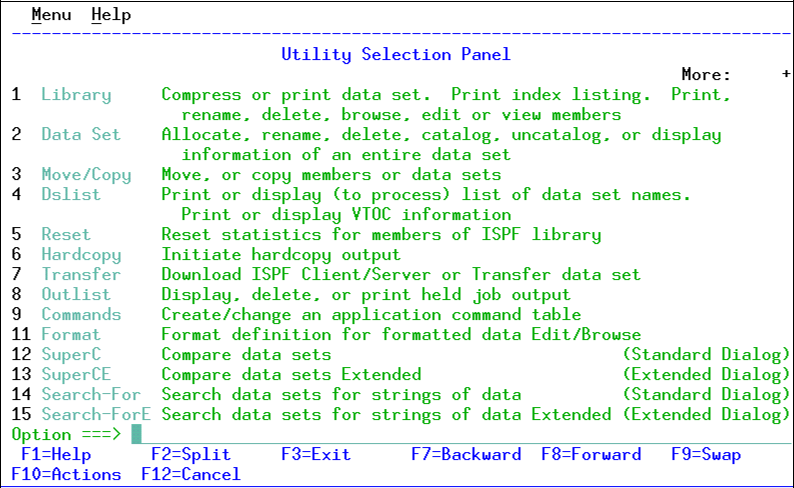


Рисунок 11 – Окно панели Utility Selection Panel.



Для того чтобы создавать, редактировать и запускать на выполнение программы, нам надо создать три набора данных (**Data Set**):

1. для хранения текстовых файлов – исходных кодов, текстов программ и др.;
2. для хранения JCL-скриптов (аналог UNIX make file), которые инструктируют z/OS о компилировании и связях исходных кодов;
3. для хранения исполняемых файлов (машинных кодов).

Сейчас мы создадим первый набор данных для тренировки, остальные наборы данных Вы создадите сами в конце лабораторной работы. Введите «**2**» – появится окно «**Data Set** **Utility**»

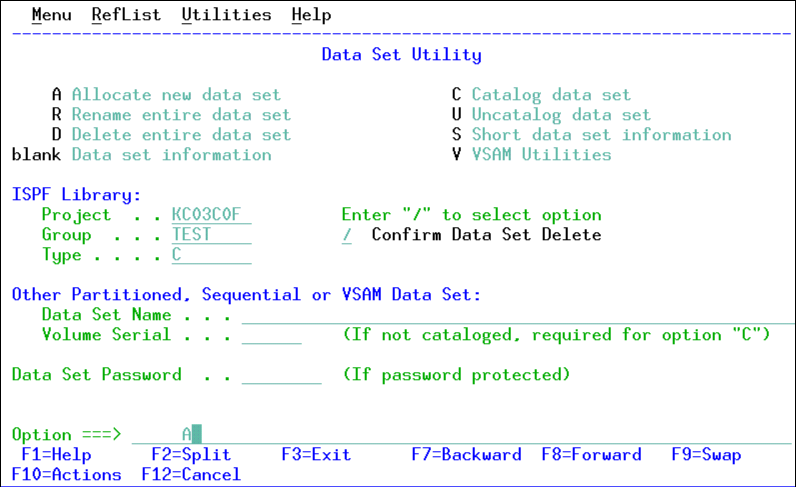


Рисунок 12 – Окно Data Set Utility.



Интерфейс **ISPF** предполагает, что мы введем имя набора данных, который хотим создать. Имена бывают простые и составные. Простое имя может содержать не более 8 символов (латинские буквы A-Z, цифры 0-9, спецсимволы #,@,$,-), причем первым символом имени не может быть цифра. Например, РАRTS01, B1934-1, $$$$A. Составное имя набора данных складывается из нескольких простых, разделенных символом "." ("точка"). Например, D.USER1.JCL, А.VERY.LONG.DATASET.NАМЕ, $PARTS.DАTА2. Имя файла-раздела формируется по тем же правилам и указывается в круглых скобках после имени библиотечного набора данных, например: MY.DSET.PROG(PROG01) - раздел PROG01 набора данных MY.DSET.PROG. Для начала создадим наш набор данных с именем в формате xxx.yyy.zzz.

PROJECT xxx (Ваш UserID)

GROUP TEST

TYPE C

* **этом материале для создания файлов используется пользовательский UserID KC03С0F. При выполнении лабораторной работы вместо него вы должны будете использовать ваш собственный UserID, выданный вам преподавателем.**
* командной строке «**Option===>**» введите «**A**» и нажмите **Enter**. В результате откроется следующее окно подсистемы распределения пространства **New Data Set Allocation** (рисунок 13).

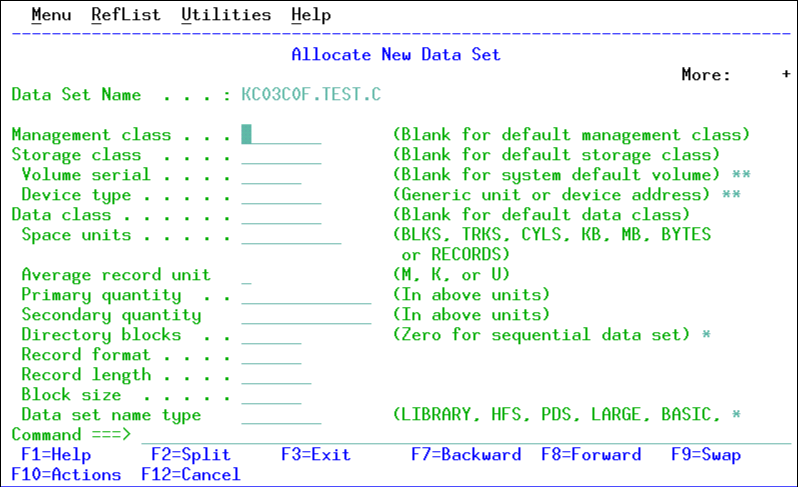


Рисунок 13– Окно определения нового набора данных.



Мы хотим создать набор данных, в который будут помещены данные, используя редактор. Мы выбрали имя набора данных «**xxx.TEST.С**» и ввели его в три поля, как показано на рисунке 12.

**Примечание:** Бывают случаи, когда TSO и ISPF требуют, чтобы параметры быливведены заглавными буквами. Поэтому желательно всегда использовать прописные буквы при работе с TSO, ISPF и CICS.

Определим размеры нашего набора данных. Будем использовать MEGABYTE в качестве единицы измерения размера набора данных. Также могут использоваться альтернативные единицы размеров – tracks, cylinders, и другие. Мы определили максимальный размер в 2 MEGABYTE.

10

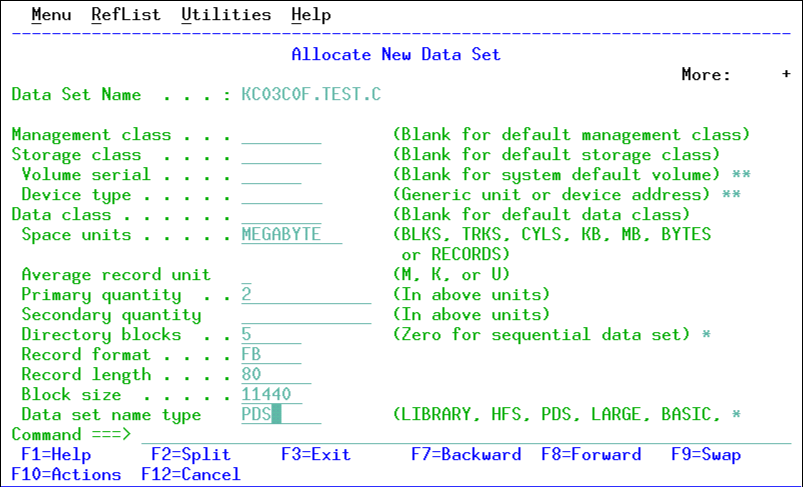


Рисунок 14 – Окно определения нового набора данных с введенными параметрами.



Тип набора данных – библиотечный (**PDS**). Для этого поставим цифру «**5**» в строке «**Directory Blocks**» (разным типам наборов данных соответствуют разные цифры - например, «**0**» будет обозначать тип «**Sequential Data Set**», «**5**» - «**Partitioned Data Set**»). Также непосредственно укажем тип набора данных в поле «**Data Set Name Type**» - «**PDS**»

Немного о структуре наборов данных. В z/OS набор данных представляется в виде совокупности логических записей, а приложения получают доступ к логическим записям и обрабатывают их как единое целое. В то же время обмен данными между периферийными устройствами и основной памятью (ввод-вывод) осуществляется блоками (или физическими записями). В блоке объединяется некоторое количество логических записей. Таким образом, для каждого набора данных необходимо установить согласованные размеры логических записей (LRECL) и блоков (BLKSIZE). То есть, размер блока должен быть получен умножением длины записи на определенное число, например: 80 х 143 = 11440.

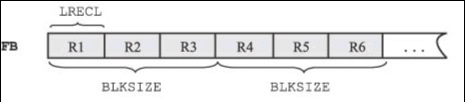


Рисунок 15 – Набор данных с записями фиксированной длины.



* z/OS поддерживаются три формата логических записей: записи фиксированной длины, записи переменной длины, записи неопределенной длины. Записи фиксированной длины имеют постоянный размер и идентифицируются символами F или FB в зависимости от выбранного способа блокирования записей: F - в каждом блоке содержится только одна логическая запись, FB - каждом блоке может содержаться более одной логической записи.

Формат записи (**Record**) определим как «**FB**» (**Fixed Blocks** – фиксированные блоки), длина записи (**Record Length**) – «**80 Bytes**» и размер блока – **11440** записей. Выбор длины записи в 80 байт не случаен – он оптимален для представления текстовых файлов, так как это длина строки в терминале и на перфокарте :) Остальные поля заполняются автоматически.

Введите параметры нового набора данных и нажмите **Enter**.

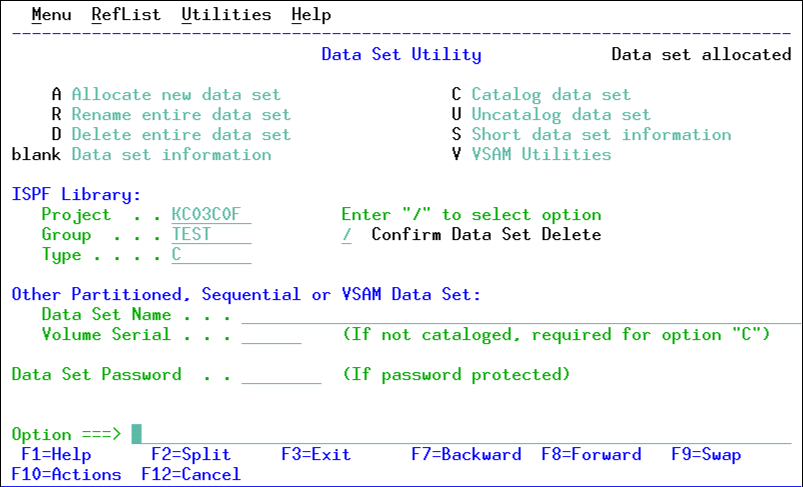


Рисунок 16 – Новый набор данных создан (allocated).



* правом верхнем углу (рисунок 16) вы увидите сообщение о том, что новый набор данных был определен (**allocated**). Наш набор данных способен теперь включить в себя несколько файлов – разделов (**members**). Нажмите F3 и вернитесь в панель «**Utility Selection** **Panel**» (см. рисунок 17).

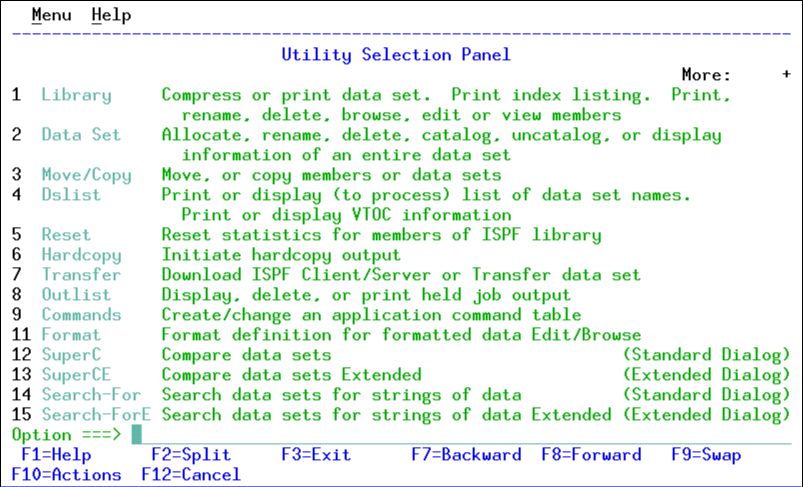


Рисунок 17 – Панель Utility Selection Panel.



Давайте проверим результат нашей работы. Это можно сделать с помощью функции **Dslist** (**Data Set List**), введите цифру «**4**» в командной строке и нажмите **Enter** (см. рисунок 17).



Рисунок 18 – Окно Data Set List Utility.

Окно «**Data Set List Utility**» предоставляет много возможностей по просмотру и редактированию наборов данных.

Так как в поле **Dsname Level** уже должен быть ваш UserID, просто нажмите **Enter**. Выведется полный список пользовательских наборов данных – Вы увидите созданный Вами набор данных (см. рисунок 19).

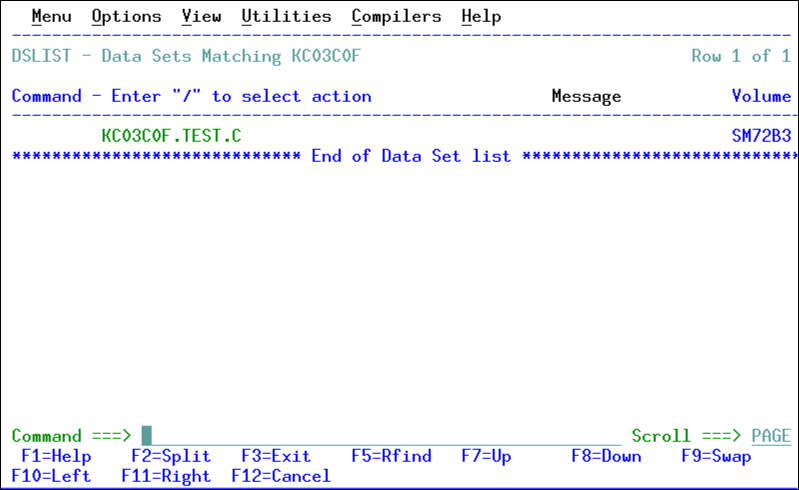


Рисунок 19 – Результат работы – созданный набор данных для файлов.

Повторите шаги и создайте еще два набора данных: **xxx.TEST.CNTL** и **xxx.TEST.LOAD**, где xxx – это ваш UserID.

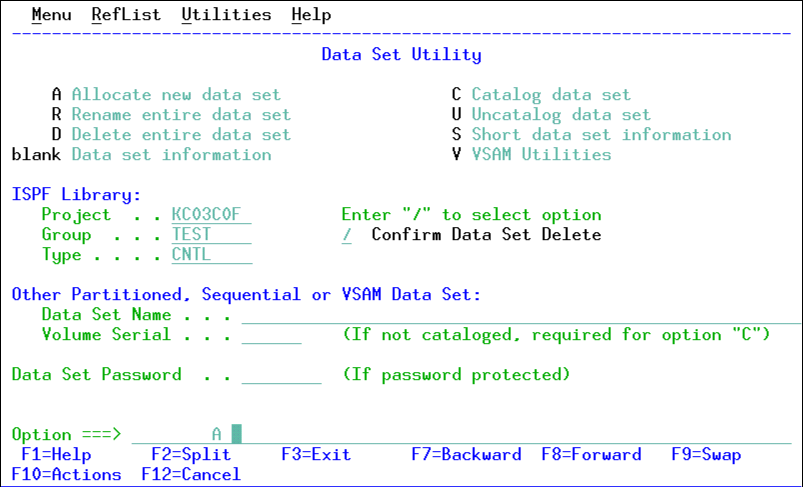


Рисунок 20– Создание набора данных xxx.TEST.CNTL.



Для создания **xxx.TEST.CNTL –** просто повторите предыдущие действия, набор данных не будет отличаться по параметрам.

Обратите внимание (см. рисунок 21) на формат записи для набора данных, в котором будут храниться выполняемые программы в машинных кодах, **xxx.TEST.LOAD** - он должен быть задан как “**U**” (c записями неопределенной длины). Отметим также, что в этом случае длину записи указывать не обязательно - при использовании записей неопределенной длины (формат U) система не поддерживает деления набора данных на логические записи и производит его обработку блоками фиксированного размера.

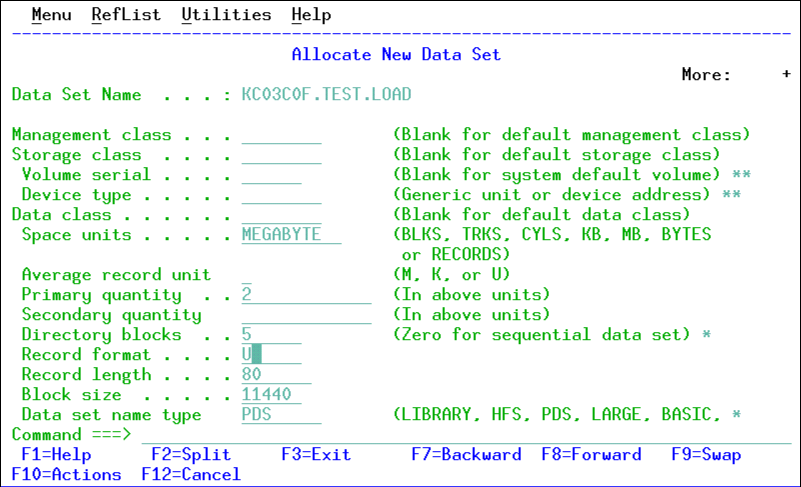


Рисунок 21 – Окно определения нового контейнера (набора данных) xxx.TEST.LOAD с введенными параметрами.

После создания еще двух контейнеров xxx.TEST.LOAD и xxx.TEST.CNTL результат вашей работы будет выглядеть так, как показано на рисунке 22.

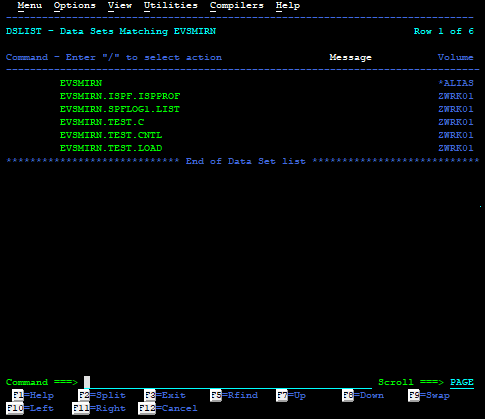


Рисунок 22 – Результат запроса DSLIST.

***4 Выход из операционной системы z/OS (z/OS Logoff)***

Нажмите несколько раз клавишу **F3 – шаг назад**, чтобы вернуться в основное окно ISPF. Очередное нажатие клавиши **F3** приведет вас в окно «**Specify Disposition of Data Set**». Система хочет знать, что вы собираетесь сделать с теми наборами данных, в которых хранится журнал событий, и предоставляет вам на выбор список действий (см. рисунок 23). Выберите «**3**».

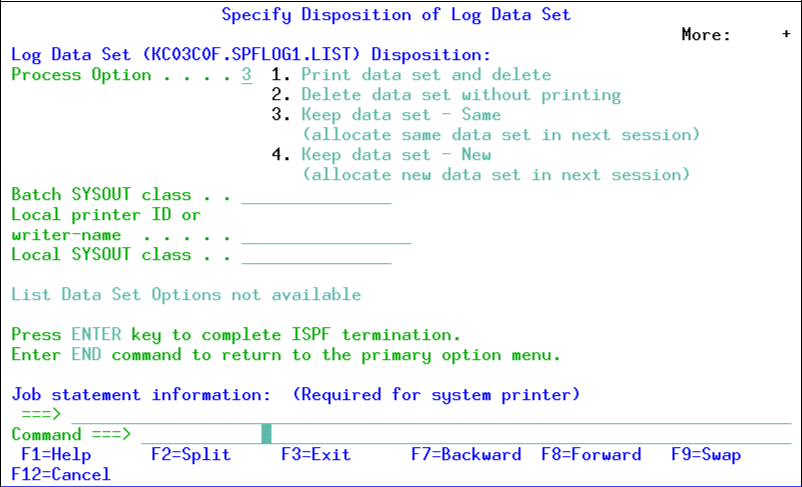


Рисунок 23 – Окно Specify Disposition of Data Set.

Произойдет сохранение данных. Появится сообщение TSO о том, что созданный набор данных будет доступен для работы при следующем обращении к системе (см. рисунок 24). При

этом набор данных **xxx.SPFLOG1.LIST**, на который ссылается система, был создан ею автоматически.



Рисунок 24 – Сообщение TSO.



Сообщение «**READY**» означает ожидание команды от пользователя. Введите слово «**LOGOFF**» и нажмите **Enter**.

**Важно:** Если этого не сделать, а просто закрыть окно эмулятора, сессия останетсянезавершенной, и вы не сможете зайти в систему, т.к. формально вы все еще находитесь в ней.

* этом случае придется подождать 30 минут. По истечению таймаута вы снова сможете зайти в систему

Эмулятор 3270 вывел вас опять в окно приветствия операционной системы **z/OS**, ожидающее от вас команд.

На этом вы можете закончить сессию связи с виртуальной средой большой вычислительной машины MAINFRAME.

ПОЗДРАВЛЯЕМ ВАС! ВЫ СДЕЛАЛИ ПЕРВЫЙ ШАГ В НОВУЮ СРЕДУ!

***5 Контрольные вопросы***

5.1 Какие способы обращения к операционной среде z/OS вы знаете?

5.2 Что такое эмулятор 3270?

5.3 Что такое TSO?

5.4 Что такое ISPF?

5.5 Перечислите основные функции ISPF.

5.6 Какую структуру имеют подсистемы и функции ISPF?

5.7 Перечислите последовательность шагов по созданию набора данных.

5.8 Какие типы наборов данных вы знаете?

5.9 Что такое «каталог»?

5.10Какие поля характеристик набора данных заполняются операционной системой автоматически, и в каких случаях,

5.11Какая функция ISPF отвечает за просмотр наборов данных?

5.12Как выйти из режима ISPF?

Министерство науки и высшего образования Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления» Кафедра «Компьютерные системы и сети»

Утверждаю

Зав.каф. ИУ6

\_\_\_\_\_\_\_\_\_\_Пролетарский А.В.

«\_\_\_»\_\_\_\_\_\_\_ 2020 г.

Галямова Е.В., Егоров М.А., Сергеев В.В.

Лабораторная работа № 2

**Редактирование программ в среде Mainframe с**

**использованием редактора ISPF**

Методические рекомендации по курсу «Архитектура больших ЭВМ»

2020 год,

Москва

**Лабораторная работа №2**

Редактирование программ в среде MAINFRAME c использованием редактора ISPF.

Цели лабораторной работы:

освоение навыков редактирования разделов набора данных с использованием интерфейса редактирования ISPF;

создание программы на С в редакторе ISPF; компиляция ее с помощью скрипта JCL,

выполнение созданной программы, вызвав программу из редактора ISPF или используя команды TSO.

**Выполнение лабораторной работы**

* **этом материале для создания файлов используется пользовательский UserID KC03С0F. При выполнении лабораторной работы вместо него вы должны будете использовать ваш собственный UserID, выданный вам преподавателем.**



***1Вход в операционную систему z/OS***

Войдите в TSO и вызовите окно интерфейса ISPF (см. рисунок 1).

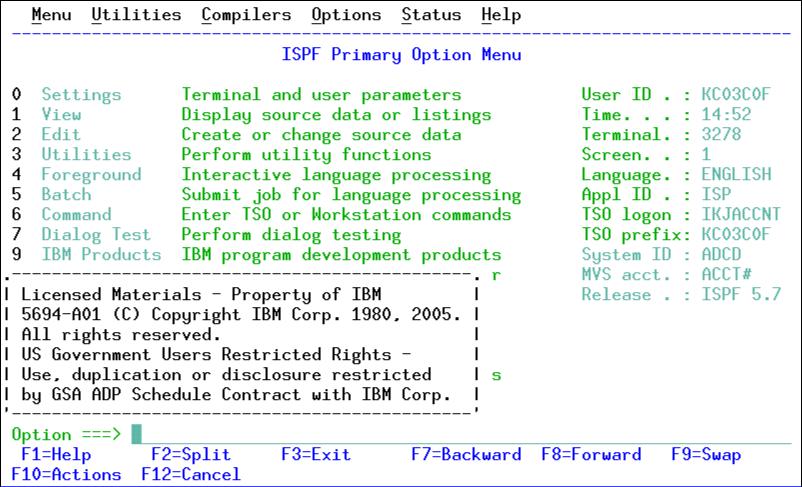


Рисунок 1 – Начальное окно интерфейса ISPF.

***2 Подготовка к разработке программы***

После выполнения лабораторной работы №1 у вас должно быть создано 3 набора данных:

xxx.TEST.С – для хранения программ в исходных кодах;

xxx.TEST.СNTL – для хранения компилированных программ;

xxx.TEST.LOAD – для хранения программ в машинных кодах (выполняемых).

где **xxx** – ваш UserID

Если какой-либо из наборов данных отсутствует, необходимо создать его способом, описанным в лабораторной работе №1.

***3 Редактирование данных в режиме ISPF Editor (кодирование на языке С)***

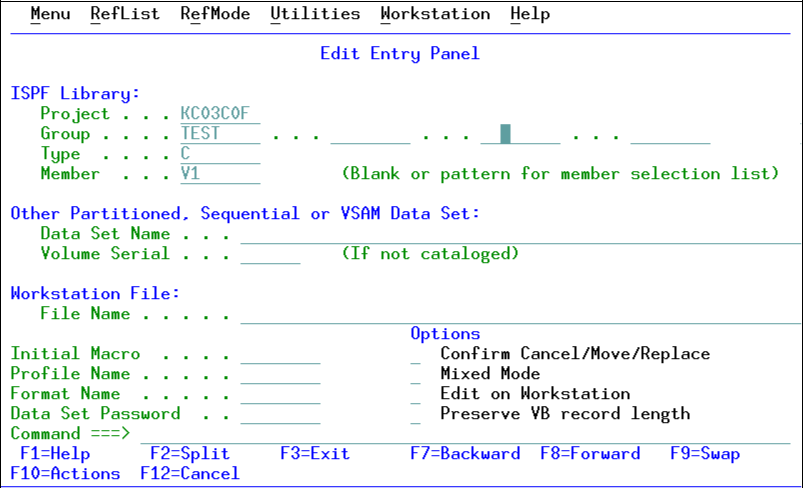
Вызвать режим редактирования вы можете из основного экрана **ISPF**, введя «2». Давайте создадим программу на языке С, используя редактор **ISPF**. В окне подпрограммы редактирования **ISPF** вы должны ввести имя файла-раздела (**Member**), который будет хранить исходный код программы. Исходный код программы будет храниться в библиотечном наборе данных **xxx.TEST.С**. Раздел, в котором хранится код, должен иметь имя - назовем его **V1** (Version 1). Полное имя набора данных - **xxx.TEST.С(V1)**. Введите эти значения в соответствующих полях, как показано на рисунке 2 и нажмите **Enter**.

Рисунок 2 – Окно входа в режим редактирования Edit Entry Panel.

После того, как будет выделена память для нового набора данных, экран редактирования будет иметь вид, показанный на рисунке 3.

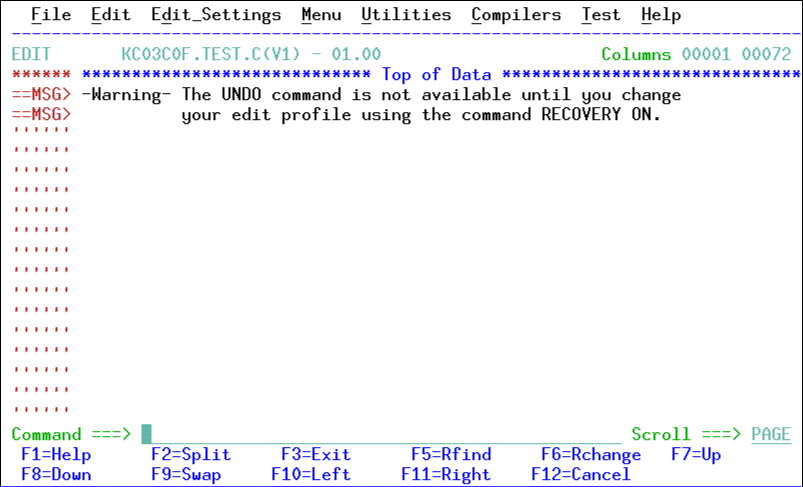


Рисунок 3 – Пустое окно редактирования ISPF.

Используйте цифробуквенные клавиши, чтобы ввести текст, и не пользуйтесь никакими управляющими клавишами, кроме клавиши **Del**. Курсор можно двигать клавишами со стрелками.

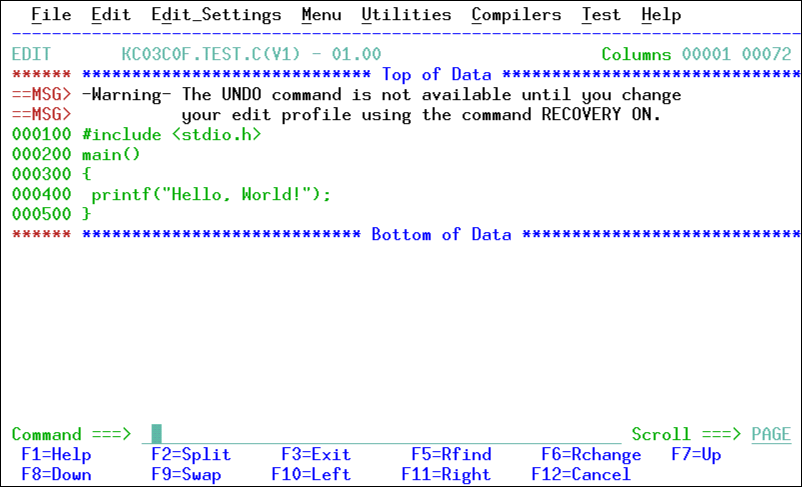


Рисунок 4 – Окно подпрограммы редактирования ISPF с набранной программой на С.

Чтобы вставить строку, поставьте курсор в левое поле (область номеров строк), в месте, где вы хотите вставить пустую строку, и введите команду **Ixx (Insert)**, где xx – количество строк, которые нужно вставить. Если понадобится удалить строку, введите в левом поле **D** **(Delete)** и нажмите **Enter.** Нажав клавишу **F3**, вы выйдете из редактора с сохранением файла.Наберите программу, как показано на рисунке 4. Разместите символы точно, как показано. Наша программа предназначена для вывода текстовой строки на экран. Затем нажмите **F3** для возврата в главное окно. Программа будет автоматически сохранена.

**Внимание:** Все ключевые слова в коде программы должны быть написаны **строчными** буквами. Может так случиться, что редактор ISPF автоматически конвертирует эти буквы в заглавные. Если это произойдет, введите в командной строке экрана, показанного на рисунке 4, команду TSO «**CAPS OFF**», затем нажмите **Enter**. В правом верхнем углу на экране в окне входа в подпрограмму редактирования ISPF после написания исходных кодов появится надпись о том, что наш файл был сохранен.

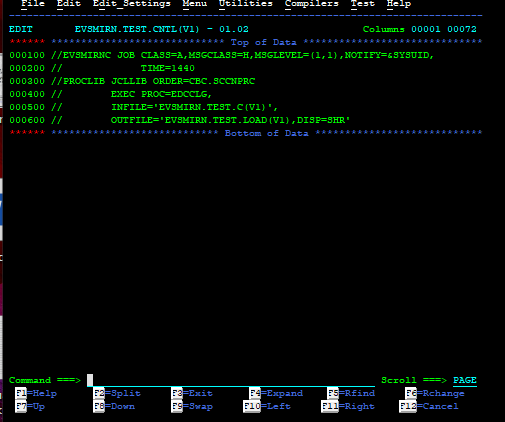
**ВАЖНО! Для освоения функций редактирования в редакторе ISPF вы должны получить у преподавателя учебный курс в электронном виде (на CD-ROMe), изучить его и пройти автоматическое тестирование по этому курсу.**

***4 Создание JCL-скрипта и его выполнение***

Пользователь z/OS может запросить у системы выполнение какой-либо работы (конечно, связанной с запуском определенных приложений) с помощью специальным образом записанного и переданного системе текста. Этот текст - задание. Задания составляются на языке управления заданиями **JCL (Job Control Language)** и направляются в систему пользователями через входные устройства и сетевые коммуникации, а также через ранее запущенные приложения. В задании указано, какие программы, в какой последовательности и какими данными должны быть исполнены, а также в какой форме и куда должны быть направлены результаты выполнения программ.

Обычно текст задания создается в некотором наборе данных с помощью текстового редактора, откуда специальными средствами пользовательского интерфейса его можно направить на обработку в подсистему JES2. Система z/OS требует, чтобы для представления заданий использовались только последовательные и библиотечные наборы данных с параметрами FB и длиной записи 80. Создадим задание, отправляющее нашу программу на компиляцию, как файл-раздел (member) в наборе данных формата PDS (Partitioned Data Set) **xxx.TEST.СNTL(V1)**.

Создайте еще один файл-раздел **xxx.TEST.CNTL(V1)**. Введите «**CNTL**» в поле «**Type**» и «**V1**» в поле «**Member**». Нажмите **Enter**. Вы попадете в окно редактора **ISPF**. Введите текст скрипта.



Замените UserID, который приведен в тексте скрипта на свой. Символ «С» в первой строке скрипта, идущий после UserID без пробела, должен следовать и за вашим UserID, он необходим для идентификации типа работы в планировщике z/OS.

Важно соблюдать расстояния между словами так, как это показано в примере на рисунке 5, т.к. у редактора есть свои «зоны» распознавания исходного текста.

Немного слов о структуре скрипта. Задание состоит из последовательности управляющих предложений JCL. Каждое предложение имеет следующую структуру:

//ИМЯ ОПЕРАТОР ОПЕРАНДЫ КОММЕНТАРИЙ

* + первых двух позициях всегда указываются две косые черты, которые являются главным отличительным признаком предложений JCL. Поле «имя» начинается с третьей позиции и служит для идентификации представленного в предложении оператора. В нашем примере такие идентификаторы – KC03C0FС, PROCLIB, CCL. Фактически это метки, на которые можно ссылаться из различных предложений задания или других заданий. Имя может содержать не более восьми символов, включающих латинские буквы, цифры и специальные знаки ($ # @) и должно начинаться с буквы или специального знака.

Оператор JOB задает начало задания и режим выполнения задания. Параметр CLASS (класс) относит задание к определенному классу выполнения заданий и задается символами A-Z, 0-9. Параметр MSGCLASS (класс сообщений) определяет выходной класс для системных сообщений, формируемых в процессе выполнения задания. Класс задается символами A-Z, 0-9. Параметр MSGCLASS дает возможность выводить все системные сообщения и выходные наборы данных, формируемые в шагах задания, в один и тот же класс или в разные классы. Такое разделение иногда может оказаться полезным. Параметр MSGLEVEL (уровень полноты сообщений) определяет, какую информацию необходимо выдать в отчет о выполнении задания. MSGLEVEL=(1,1) означает максимально подробный вывод отчета. Параметр TIME (время) устанавливает максимальную продолжительность выполнения задания в минутах.

Оператор JCLLIB задает cписок библиотек для поиска процедур, указанных в задании.

* нашем примере используется библиотека компиляторов CBC.SCCNPRC.

Оператор EXEC указывает выполняемую программу или процедуру и ее параметры. В нашем примере вызывается процедура компиляции программы на языке C – EDCCLG. В ее параметрах – входной набор данных. содержащий текст программы на С и выходной набор данных, в котором будет содержаться полученная программа. Параметр DISP (диспозиция) определяет исходное состояние выходного набора данных, а также действия, которые необходимо произвести с набором данных после завершения шага задания или всего задания: сохранить, уничтожить, каталогизировать и др. DISP=SHR означает, что набор данных уже существует и может быть использован одновременно другим заданием, т.е. разделяется различными заданиями в режиме чтения.

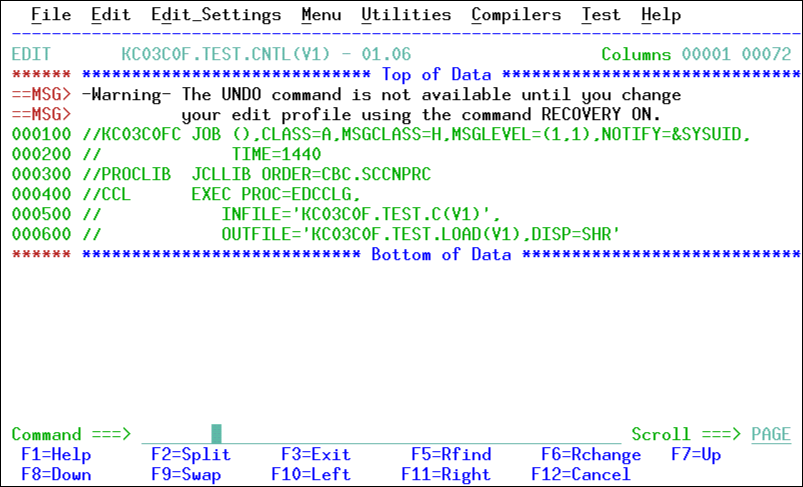


Рисунок 5 – Окно подпрограммы редактирования ISPF с набранным скриптом JCL.

При изменении и корректировке файла не забывайте сохранять изменения клавишей F3 (по ее нажатию вы выйдете из редактора, вам нужно будет зайти еще раз).

Теперь необходимо отправить задание на исполнение. Задания, поступающие в систему от различных источников, принимаются и обрабатываются специальным компонентом z/OS, который называется подсистемой управления заданиями **JES (Job Entry Subsystem). JES** принимает задания, поступающие с входных устройств, регистрирует их, осуществляет анализ

* формирует очереди заданий, а затем передает задания на выполнение базовой управляющей программе BCP. После завершения выполнения задания и получения результатов от BCP, JES формирует отчет по заданию (листинг), передает его пользователю или выводит на указанные устройства. Убедитесь, что Ваше задание отправлено на выполнение и находится в очереди заданий. Сделайте скриншот этого экрана.

***5 Создание программы на языке REXX и ее выполнение***

Повторите действия по созданию нового раздела в наборе данных и напишите первую программу на языке REXX, как показано на рисунке 6. Сохраните ее , нажав F3.

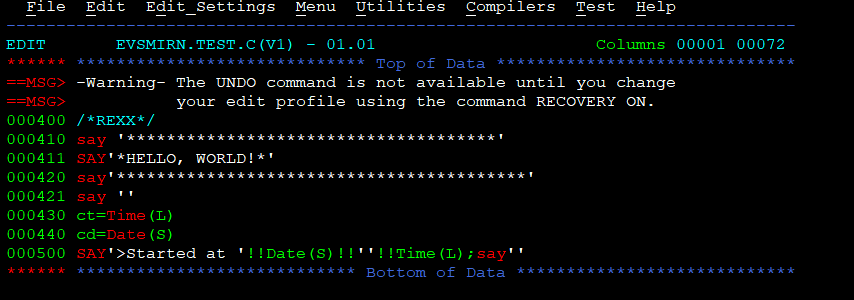


Рисунок 6. Программа на языке REXX

Для запуска задания на выполнение, находясь в редакторе скрипта, в поле Command введите команду SUB или SUBMIT. Либо в режиме редактирования раздела в левом столбце введите команду EXEC на выполнение, как показано на рисунке 7.

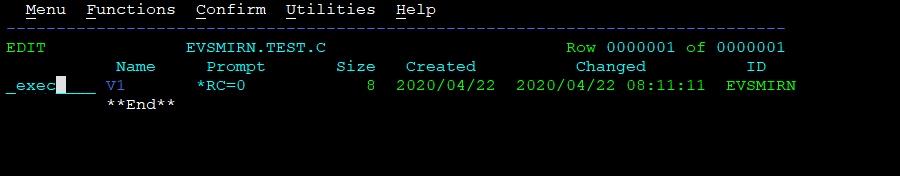


Рисунок 7.

Если задание отправлено в очередь, то появится сообщение (рисунок 8).





Рисунок 8.

Вы должны увидеть сообщение об отправке задания. Знак «\*\*\*» - значит, что для просмотра остатка сообщения необходимо нажать кнопку **Enter**. Если скрипт и программа ошибок не содержат, то вы получите сообщение об успехе (**MAXCC** равен **0**), как показано на рисунке 9.



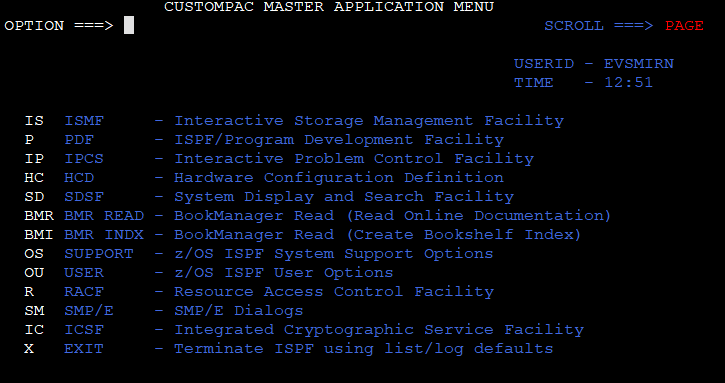
Рисунок 9

* противном случае вам придется искать ошибку в скрипте или программе.

***5 Поиск ошибок и работа с утилитой SDSF***

Для того, чтобы посмотреть отчет об ошибке, можно просмотреть журнал заданий через **утилиту SDSF.**

Запустить **утилиту SDSF** можно из среды TSO в интерфейсе



Рисунок

Для этого необходимо знать имя задания и UserID. Утилита SDSF выведет список последних запущенных заданий. Установив фильтр, можно быстро найти интересующее нас задание.

Также можно воспользоваться функцией Outlist в ISPF, для этого нужно знать имя задания и идентификационный номер задания. Имя задания задается в первой строке скрипта JCL. Зачастую это UserID и постфикс, в нашем случае - «С», означающий compile – компиляция. Идентификационный номер задания – JobID – можно видеть в отчете об отправке задания, сразу после ввода команды Submit.

Теперь после компиляции двоичный код нашей программы содержится в файле **xxx.TEST.LOAD(V1)**. Давайте запустим программу. Для этого нам необходимо ввести в полеCommand:

TSO CALL 'KC03C0F.TEST.LOAD(V1)'

Вместо KC03C0F введите свой UserID. После этого вы должны видеть результат работы вашей программы на экране.

***5 Выход из операционной системы z/OS (z/OS Logoff)***

Нажмите несколько раз клавишу F3, чтобы вернуться в основное окно ISPF. Очередное нажатие клавиши F3 приведет вас в окно «**Specify Disposition of Data Set**». Система хочет знать, что вы собираетесь сделать с наборами данных лог-файла, которые вы создали, и предоставляет вам на выбор список действий (рисунок 6). Выберите «3».

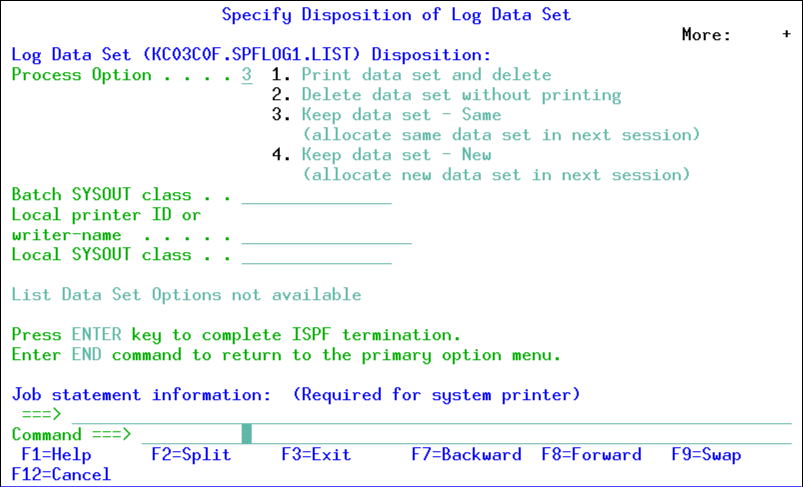


Рисунок 6 – Окно Specify Disposition of Data Set.



Сообщение **TSO** о том, что созданный набор данных будет доступен для работы при следующем вашем обращении к системе. При этом набор данных **xxx.SPFLOG1.LIST**, на который ссылается система, был создан ею автоматически.

Затем введите «**LOGOFF**» и нажмите **Enter**.

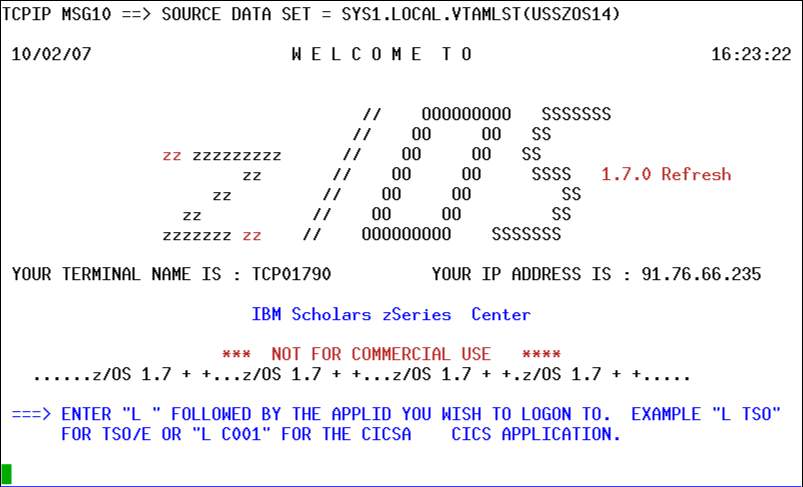


Рисунок 7 – Окно завершения работы в ISPF.

Эмулятор 3270 вывел вас опять во окно приветствия операционной системы z/OS, ожидающее команд. На этом вы можете закончить сессию связи с виртуальной средой большой вычислительной машины MAINFRAME.

1. ***Контрольные вопросы***

6.1 Из каких компонентов состояла среда разработки, которую Вы использовали в этой работе? Перечислите инструменты, использовавшиеся в лабораторной работе.

6.2 Опишите процесс создания программы на языке С в среде mainframe?

6.3 Что такое JCL? Что такое JES?

6.4 Каким образом можно скомпилировать готовую программу на языке С в среде mainframe?

6.5 Перечислите основные операторы языка JCL и опишите их назначение.

6.6 Как отправить скрипт JCL на обработку в среду JES?

6.7 Почему тип набора данных xxx.TEST.LOAD был U – undefined?

6.8 Что необходимо сделать, если скрипт JCL вернул ошибку?

6.9 Как запустить скомпилированную программу в среде mainframe?

6.10 В чем разница между компиляцией программы на языке С и программы на другом языке среды mainframe, например COBOL?

6.11 Какие системные библиотеки использовались в JCL скрипте и почему?