РАЗРАБОТКА IO ФУНКЦИЙ ДЛЯ НАБОРОВ ДАННЫХ ОПЕРАЦИОННОЙ СИСТЕМЫ Z/OS

Шатов А. М., студент кафедры "Информатика и Информационная Безопасность" ПГУПС Императора Александра I,
alexsandr.shatov@yandex.ru,

Шатов В. М студент кафедры "Информатика и Информационная Безопасность" ПГУПС Императора Александра I,
valentin.shatov@yandex.ru

Аннотация

В работе описаны аспекты разработки функций ввода/вывода для наборов данных операционной системы z/OS, текущий результат и отмечены направления дальнейшего развития API.

Введение

Существуют программы, в частности системы управления базами данных (СУБД), которые имеют встроенную в них логику "зависимых записей" для того, чтобы гарантировать целостность данных в случае аварийных ситуаций на узле, в программном обеспечении или в подсистеме хранения данных. .

“Зависимой записью” является такая запись, которая не будет выполнена, пока не завершится предыдущая запись.

Примером является обновление базы данных. Когда СУБД обновляет базу данных, она в первую очередь делает запись на диск, содержащий системный журнал (лог). Затем данные записываются в базу данных и, в конце концов, вновь в системный журнал для того, чтобы указать, что обновление было выполнено. Все эти три операции ввода/вывода (лог, база данных, лог) связанны и последующая операция не будет выполнена, пока предыдущая не закончится успешно.

Согласованность данных. IVP

Если данные находятся в том порядке, в котором они должны быть и, если нет пропущенных данных, то говорят, что данные согласованны (data is consistent).

В системах удалённого копирования данных, согласованность данных (data consistency) не может быть гарантирована, если операция ввода/ввода была удалённо отзеркалирована, а предшествующая ей операция нет. Для

того, чтобы быть уверенным в том, что ПО удалённого копирования данных работает должным образом и данные на резервных дисках согласованны, было разработано приложение IVP (Installation Validation Program for data consistency), которое позволяет проверить на резервных дисках согласованность скопированных данных.

Функции IVP:

1. Генерирование «зависимых записей» и их запись на устройства хранения данных;

2. Проверка данных на согласованность.

На данный момент IVP имеет ряд недостатков:

1. не поддерживает работу с Linear и RRDS VSAM наборами данных. Кроме того, разработчику необходимо учитывать особенности работы API с каждым из типов наборов данных;
2. при работе с большим объёмом данных работает относительно медленно, так как функции ввода/вывода написанные на языке C используют Language Environment сервис для взаимодействия с системными функциями ввода/вывода;
3. При обнаружении несогласованности, IVP возвращает смещение, считанной на момент обнаружения записи (номер записи), относительно набора данных, в котором данная запись находится. Удобнее, чтобы IVP могло возвращать не смещение записи, а номер цилиндра и номер трека, где эта запись хранится или куда записывается.

Разработка API

Постановка задачи

Необходимо разработать специализированное API, расширяющее возможности IVP и устраняющее вышеописанные недостатки. Впоследствии оно позволит стандартизировать функции обращения к наборам данных.

Используемые языки программирования программное обеспечение

Для работы и тестирования создаваемых программных модулей на языке HLASM используется эмулятор Hercules (см. Рис. 1) с установленной на нём z/OS, т. к. участники Co-op проекта не имеют доступа к реальным мейнфреймам.

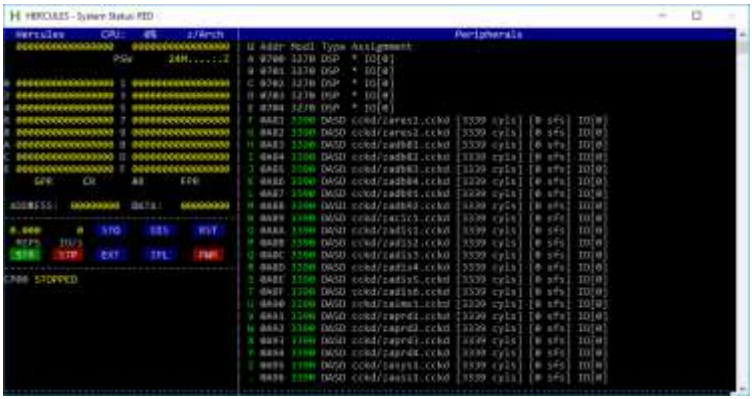

Рисунок 1: Интерфейс эмулятора Hercules

Подключение к эмулятору производится при помощи терминала Vista TN3470 (см. Рис. 2). Промежуточные результаты разработок предоставляются раз в неделю, непосредственно работа над модулями проводится дома в свободное время.

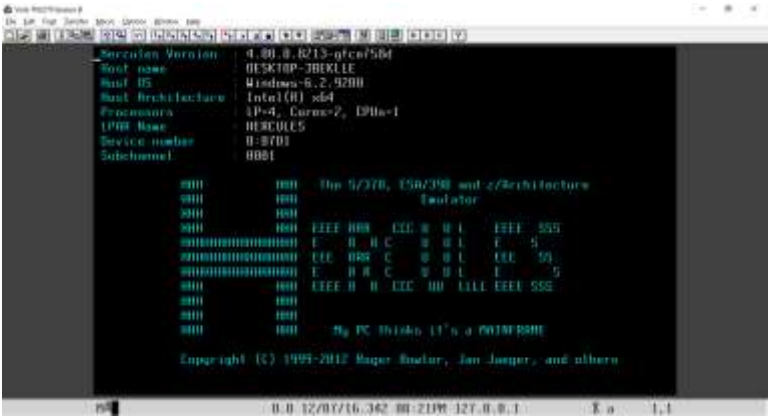

Рисунок 2: Интерфейс терминала Vista TN3270

Текущие результаты

На данный момент нами разработаны первые версии ассемблерных модулей, реализующих:

1. работу с участком виртуальной памяти в дальнейшем

интерпретируемым как дескриптор набора данных (E_FILE);

2. операции ввода/вывода с PS, DA и LDS наборами данных;

3. статический и динамический вызовы подпрограмм;

4. разбор входной строки с опциями открытия набора данных и определяющий его тип (PS, PO, DA, Indexed Sequential Organization, VSAM).

Также был разработан модуль на языке программирования C, выполняющий тестирование корректности выполнения операций ввода/вывода с PS, PDS и DA наборами данных. Кроме того в процессе разработки находятся модули для работы с ESDS, KSDS, DA, PS/PDS наборами данных.

Дальнейшее развитие проекта

Для создания законченного, работающего API, поддерживающего работу с требуемыми типами наборов данных и методами доступа к ним, необходимо:

1. Добавить возможность определения недостающих типов наборов данных;

2. Написать ассемблерные модули для работы с недостающими типами;

3. Реализовать возможность определения абсолютного адреса (цилиндр, трек) записи, считанной с диска, для всех типов наборов данных;

4. На языке программирования C написать модули, выполняющие тестирование функций ввода/вывода всех требуемых типов наборов данных.

Заключение

В рамках данной работы были разработаны подули для взаимодействия с PS, DA и LDS наборами данных. Кроме того получены навыки использования подсистем z/OS (ISPF, SDSF, IDCAMS и т. д.). Кроме того, было произведено тестирование созданных модулей.

Литература

1. IBM z/Architecture Principles of Operation — 2015.
2. High Level Assembler for z/OS & z/VM & z/VSE. Toolkit Feature User's Guide – 2013.
3. DFSMS: Using Data Sets – 2005.