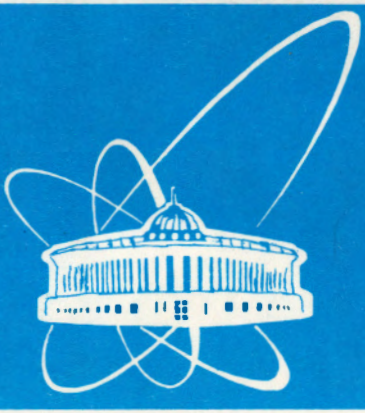

сообщения
объединенного
института
ядерных
исследований
дубна

Р10-94-151

А.С.Кирилов

МЕТОДИКА ОРГАНИЗАЦИИ УПРОЩЕННОГО ДОСТУПА К УСТРОЙСТВАМ В ОПЕРАЦИОННОЙ СИСТЕМЕ OS9 И СРЕДСТВА ЕЕ РЕАЛИЗАЦИИ

1994

Введение

OS9 - многозадачная, многопользовательская операционная система реального времени. В отличие от однозадачных операционных систем, таких как MS-DOS, в целях взаимозащиты задач и повышения общей надежности в ней проведено четкое разделение сфер системной (управление процессами, системой прерываний, распределением памяти, вводом-выводом и т.д.) и пользовательской деятельности, которое обеспечено как аппаратными, так и программными средствами. В частности, программа обычного пользователя не может прямо обратиться к произвольному участку адресного пространства в условиях действия системы защиты памяти или зарегистрировать процедуру обслуживания прерывания (ПОП). Поэтому согласно схеме организации ввода вывода, принятой в системе, для программного обслуживания каждого нового электронного блока (контроллера и т.д.) должен составляться специальный драйвер, и, если не пригоден ни один из существующих, еще и менеджер файлов (file manager). В том случае, если вновь подключаемый блок весьма прост в управлении, например в сравнении с контроллером диска или графического устройства, такая перспектива не вызывает энтузиазма, тем более, что и менеджер файлов ввиду его сложности не рекомендуют писать даже специалисты в этой области /1/.

Поэтому во многих случаях предпочитают воспользоваться различными обходными методами, которые предусмотрены в системе для нестандартных ситуаций.

"Преодолеть" противодействие системы защиты памяти можно разными способами, в том числе:

а) отменой защиты памяти;

б) открытием области адресов регистров устройства с помощью F$Permit;

в) тем же,включив указанную область в список цветной памяти (colored memory) OS9;

г) созданием привилегированной задачи;

д) введением дополнительного системного вызова;

е) составлением системного трапхендлера (trap handler).

Рассмотрим характеристики этих способов. Способ а) с одной

стороны, позволяет получить самый быстрый и простой доступ из программы на языке высокого уровня к устройству (см. раздел "Временные характеристики"), а с другой стороны, приводит к ситуации, когда все задачи в системе получают неконтролируемый доступ ко всему адресному пространству, что чревато непредсказуемыми последствиями в случае ошибок в любой из них. Поэтому он популярен при наладке вновь разработанных блоков, но едва ли уместен вне этой сферы. Способы б) и в) более применимы для подключения блоков памяти. Способ г) доступен только для привилегированного пользователя (super user) и обладает основным недостатком способа а). Способы д) и е) по существу одинаковы, но второй представляется более удобным.

Для подключения ПОП применимы способы г)-е).

Методика Й.О.Петерсена и П.Шарфф-Хансена

В работе/2/ описана методика подключения контроллера КАМАК к VME-системе под управлением OS9. Для этого была использована комбинация способов г) и ж), а именно:

- для доступа к регистрам КАМАК составлен системный трапхендлер ;
- для обслуживания прерываний - упрощенный драйвер вместе с SCF-дескриптором;
- для интерфейса с программой пользователя - Си-доступная библиотека процедур работы с КАМАК, соответствующая требованиям ESONE.

Связь ПОП и программы пользователя организована посредством событий (events), причем для каждого вектора прерываний создано отдельное событие.

Преимущество подобного подхода по сравнению с драйвером заключается не только в превосходстве в скорости взаимодействия с КАМАК. Очень важно также, что конечному пользователю предоставлена возможность самостоятельно, редактируя дескриптор, изменять базовый адрес крейта, номер вектора и уровень прерывания - то есть при необходимости подстроить все под параметры своей аппаратуры. Более того, в связи с тем, что драйвер сильно упрощен и фактически состоит из процедуры INIT, используемой для подключения ПОП, и собственно ПОП, он не требует установки бита SupStat в байте атрибутов, и, следовательно, может модифицироваться и т.п. обычным пользователем.

Подключение ПОП производится с помощью команды инициирования устройства INIZ, которая, в частности, вызывает процедуру INIT. Кроме всего прочего после выполнения команды INIZ данное устройство действительно включается системой в список своих устройств. Поэтому по команде DEVS, например, можно убедиться в корректности парамеров, заданных в дескрипторе.

Хотелось бы отметить и следующее важное на взгляд автора методическое достоинство подхода. Для настройки всего комплекса программ в целом пользователь вносит изменения только в дескриптор, что сравнительно просто и не требует специальных системных знаний.

Подобная методика была применена автором данной статьи для работы с интерфейсным блоком VME-KK009 и рядом других блоков (БУЭ, БУШД) в начальном варианте программы управления спектрометром НСВР (СКАТ) /3/. При этом со всей очевидностью выявилось подобие трапхендлеров для различных блоков. Действительно, за исключением КАМАК, блоки управляются несколькими 8-16-разрядными регистрами, для которых необходимо уметь выполнять операции записи/чтения и, возможно, сброс/установку бита. Все это не могло не навести на мысль о замене индивидуальных трапхендлеров общим для всех подобных блоков.

Однако наряду с явными достоинствами методике /2/ присущ весьма серьезный недостаток. В избранном способе идентификации регистров КАМАК значением параметра, обозначающего этот регистр, является его полный 32-разрядный адрес. Поскольку сам трапхендлер не выполняет проверки корректности значений получаемых им адресов, да и при данном способе идентификации регистров не в состоянии это сделать, программа с его помощью получает неконтролируемый доступ ко всей памяти. Т.о. получается полная аналогия со снятием защиты памяти, с той лишь разницей, что это произведено для программ, использующих данный трапхендлер.

Новый подход в формировании адреса

Пока формирование адреса регистра производится в программе пользователя, трапхендлер не в состоянии нейтрализовать возможные ошибки адресации. Если же перенести эту операцию в трапхендлер, можно обеспечить как защиту, так и определенный контроль корректности адреса. Схема метода выглядит следующим образом.

Вводится новая операция регистрации устройства, во время которой базовый адрес извлекается из дескриптора устройства и наряду с именем устройства помещается в специальную таблицу во внутренней памяти трапхендлера (internal static storage). В качестве результата этой операции для идентификации устройства в программу пользователя возвращается номер устройства (соответствующий индексу занятого элемента таблицы). Конкретные регистры устройства адресуются с помощью идентификатора регистра, аккумулирующего номер устройства и смещение адреса регистра относительно базового адреса устройства. При обращении к регистру выполняется вычисление адреса регистра с соответствующими проверками, что, как показано ниже, не приводит к большим временным издержкам. Введеная процедура регистрации устройства попутно позволяет обеспечить режим монопольного доступа к устройству, что особенно актуально в многозадачной среде.

Особенности работы с прерываниями

Скорость обслуживания прерываний с помощью событий вполне удовлетворяет потребности процессов на НСВР. Поэтому эта схема из /2/ сохранена при единственном дополнении. Имя события помещается в дескриптор устройства. Оттуда оно извлекается процедурой INIT драйвера, которая в частности создает его и размещает его идентификатор также в дескрипторе. Процедура регистрации устройства передает его в программу пользователя для работы с этим событием. Таким образом имя события хранится только в дескрипторе устройства, а не в дескрипторе и программе пользователя, как в /2/, что на взгляд автора более надежно.

Следует отметить, что в случае непригодности данной схемы обслуживания прерываний по причине недостаточной ли скорости, либо по иной, возможность применения любых методов обслуживания прерывания и организации связи ПОП с основной программой в рамках предлагаемой методики ничем не ограничена.

Групповые операции чтения и записи

Существуют устройства, в которых для выполнения одной логической операции необходимо выполнить несколько физических обращений для чтения/записи. Так, например, в блоке управления

экспериментом (БУЭ)/3/, выполненного на основе программированного таймера 8254, для задания значения счетчика требуется сначала установить соответствующий режим в управляющем регистре, а затем двумя обращениями занести значение в сам счетчик. Подобные операции, как правило, находятся в так называемым "критическом" интервале, т.е. не могут быть разорваны в результате прерываний, поступивших прежде всего от данного устройства. Для выполнения этих операций предусмотрены особые средства, позволяющие за один вызов трапхендлера выполнить целый набор обращений к регистрам устройства. При этом, как это обычно делается в драйверах, этот блок обрамляется сначала запретом, а по окончании разрешением прерываний от устройств, имеющих заданный уровень прерываний.

Групповые операции могут быть применены и для ускорения обслуживания устройства, поскольку выигрыш во времени по сравнению с одиночными операциями тем существеннее, чем больше список регистров в группе.

Режим монопольного доступа к устройству

В многозадачной, особенно сетевой, среде возникает потребность в введении упорядочивания доступа к устройствам со стороны разных задач. С этой целью в дескриптор устройства было добавлено специальное поле, которое далее будет именоваться флагом занятости. Этот флаг устанавливается в момент первой регистрации устройства и сбрасывается после окончания работы с ним, означая режим монопольного доступа к устройству. Для удобства отладки программ вводится также отладочный режим работы с устройством, при котором флаг занятости не устанавливается.

Реализация

Для реализации предлагаемой методики был составлен системный трапхендлер gdt и в качестве его интефейсной части Си-доступная библиотека gdtlib. Состав библиотеки приведен в приложении. Помимо этого прилагаются тексты образцов драйвера foodrv.a и дескриптора foo0.a. В текущей версии установлены следующие ограничения:

- число одновременно обслуживаемых устройств до 16-ти;
- масимальное допускаемое смещение адреса регистра

относительно базового адреса устройства - 255;

- максимальное число операций в группе - 255.

Рисунок призван пояснить схему взаимодействия программы пользователя с устройствами при помощи gdt + gdtlib. Для работы с устройствами device0 и device1 необходимо подготовить:

- дескрипторы descr0 и descr1 на ассемблере по образцу foo0;

- драйвер driver1 на ассемблере по образцу foodrv; для device0 драйвер не требуется, т.к это устройство не генерирует прерываний;

- по желанию пользователя для большего удобства работы с устройствами библиотеки devlib0 и devlib1 на Си.

В программе пользователя при этом должны присутствовать следующие элементы:

```
    ...
int             i,
                dev0,           /* идентификатор device0 */
                dev1;           /* идентификатор device1 */
unsigned        eventID1,       /* идентификатор события */
                foo,            /* рабочая переменная    */
                dev0_READ;      /* идентификатор регистра */
char dev0name[] = "device0";    /* имя устройства        */
char dev1name[] = "device1";    /* имя устройства        */
unsigned short v;               /* значение регистра     */
    ...
i = LinkGdt();                  /* подключение gdt   к  задаче */
if (i < 0) goto error;          /* отсутствует gdt */
dev0 = LinkDev(dev0name, &foo, 1); /* регистрация устройства
                        device0; значение второго параметра не
                        существенно из-за отсутствия прерывания */
if ( dev0 < 0) goto error;      /* в случае ошибки */
dev1 = LinkDev(dev1name, &eventID1, 1);    /* регистрация
                            устройства device1 */
if ( dev1 < 0) goto error;      /* в случае ошибки */...
dev0_READ  =  DefReg(dev0,2);  /* задание регистра   dev0_READ
                    со смещением 2 относительно базового адреса */
...
v = Readr16(dev0_READ);         /* чтение регистра dev0_READ */
    ...
UnlinkDev(dev0);                /* окончание работы с device0  */
    ...
UnlinkDev(dev1);                /* окончание работы с device1 */
```



```
    ...
error:
    ...
```

Процедура LinkTrap подключает gdt к задаче. Процедура LinkDev регистрирует устройство. Она обращается к дескриптору устройства, который должен быть предварительно загружен. Если устройство работает с прерываниями, то предварительно должен быть загружен также и драйвер устройства и выполнена команда INIZ. Процедура DefReg предназначена для получения идентификатора регистра, процедура Readr16 - для чтения 16-разрядного регистра. Процедура UnlnkDev стирает информацию об устройстве из таблицы устройств gdt и делает это устройство вновь доступным для других задач. Эта операция обязательна.

Сообщение об ошибках

Для информирования программы пользователя об ошибках принят следующий подход. Процедуры LinkGdt, LinkDev и UnlinkDev возвращают код ошибки в качестве значения процедуры. Процедуры типов Readr/Writer , Read_b/Write_b и Bitsr/Bitcr при обращении к незарегистрированному устройству игнорируют операцию.

Временные характеристики

Измерение временных характеристик было выполнено на следующем оборудовании:

компьютер - CompControl (68030, 20 МГц);

устройство - БУШД - блок управления шаговыми двигателями.

При этом использовалось программное обеспечение:

ОС - OS9 v 2.4;

компилятор - Microware C compiler v 3.2.

Измерение проводилось программой, которая циклически выполняла операцию чтения (записи) 16-разрядного числа в один из регистров блока БУШД. Получены следующие результаты:

- прямое обращение через процедуру на Си (со снятой защитой памяти) - 12 мкс;

- трапхендлер в системном режиме :
  - одиночная операция по предлагаемой методике - 57-59 мкс;
  - то же с использованием адреса регистра в качестве параметра - 55 мкс;
  - без операции - 47-49 мкс;
  - блочные операции:
    (3 операции) - 77 мкс;
    (2 операции) - 73 мкс;
    (1 операция) - 69 мкс;
- драйвер SCF (без операции) - 250 мкс.

. Под прямым обращением понимается обращение к регистру через указатель, оформленное в виде процедуры на Си. "Без операции" означает время затрат на системные издержки. Как видно, для системного трапхендлера они в 5 раз меньше, чем для драйвера. В свою очередь, прямое обращение быстрее еще примерно в столько же раз. Видно также, что использование готового адреса для указания регистра существенного выигрыша по времени не дает (менее 7%). С другой стороны, применение групповых операций тем выгоднее, чем длиннее группа. Так, при ее длине, равной 3, время в рассчете на одну операцию составляет 22 мкс, что почти в три раза быстрее, чем время одиночной операции.

Заключение

В работе представлена методика организации доступа к устройствам в операционной системе OS9 в условиях действия защиты памяти. Основное достоинство методики по сравнению с драйвером состоит в простоте и скорости обслуживания устройств. Почерпнув идею из работы /2/, автор сделал следующие существенные шаги по ее развитию и дополнению:

- выбран набор операций, позволивший подготовить общий трапхендлер для работы с целым рядом устройств. В числе операций предложен и реализован набор групповых операций для выполнения действий, находящихся в "критическом" интервале;
- предложена и реализована новая (по сравнению со /2/) схема адресования регистров устройств, позволившая, во-первых, существенно повысить защиту от возможных ошибок со стороны прикладной программы как самой системы, так и других программ, причем сделать это с минимальными (5-7%) дополнительными

временными издержками, во-вторых, при необходимости вводить режим монопольного доступа к устройствам.

Предлагаются средства реализации методики в составе системного трапхендлера (gdt), интерфейсной Си-доступной библиотеки (gdtlib.l), а также вариантов драйвера (foodrv.a) и дескриптора устройства (fooO.a).

Автор выражает глубокую признательность Й.Петерсену за любезно предоставленные материалы, В.И.Приходько за помощь в работе, В.Е.Резаеву за консультации по вопросам организации аппаратуры VME, А.П.Сиротину и М.П.Коробченко за помощь при наладке программ управления контроллерами, И.Н.Чурину и А.И.Гилеву за предоставленные тексты библиотеки подпрограмм КАМАК для PC.

Литература

1. Peter Dibble. OS-9 INSIGHTS. An Advansed Programmers Guide to OS-9/68000, Microware System corporation, 1988.
2. J.O.Petersen, P.Scharff-Hansen. CAMAC Routines for OS9/VMEbus/CBD8210 Systems. CERN/ECP/DS, 1992.
3. Зен Ен Кен и др. Система накопления, управления и контроля спектрометра НСВР в стандарте VME. Препринт ОИЯИ Р13-94-73, ОИЯИ, Дубна, 1994.

Приложение. Список функций gdtlib

1) LinkGdt: подключение трапхендлера к задаче:
а) обращение:
```
int LinkGdt()
```
б) действие:
процедура подключает трапхендлер gdt к программе пользователя, возвращая значение = 0 в случае успеха или значение < 0 в случае ошибки. Эта процедура должна быть вызвана ранее других процедур gdtlib;
в) ошибка:
- 4 - неудача при подключении трапхендлера;

2) DefReg: задание регистра устройства:
а) обращение:

```
    unsigned DefReg(devID, disp)
       int devID, disp;
```

б) параметры :

```
    devID     (input)      - идентификатор устройства,
    disp      (input)      - смещение адреса регистра
                           относительно базового адреса
                           устройства;
```

в) действие:

по значениям параметров определяется значение идентификатора регистра, которое возвращается в виде значения функции;

3) LinkDev: регистрация устройства:

а) обращение:

```
    int LinkDev(devname, eventID, access_flag)
      char *devname;
      unsigned *eventID;
      int access_flag;
```

б) параметры:

```
    devname        (input)   - имя устройства,
    eventID        (output)  - идентификатор события,
    access_flag    (input)   - режим доступа к устройству
                             ( = 0 - отладочный,
                               <> 0 - монопольный );
```

в) действие:

процедура проверяет наличие свободных мест в таблице. Если таблица полна, в качестве значения процедуры выдается код ошибки. Если в таблице есть свободное место, процедура устанавливает связь с модулем devname, рассматривая его в качестве дескриптора устройства. Если устройство занято (флаг занятости в дескрипторе установлен), выдается код ошибки. Если устройство свободно, из дескриптора извлекаются базовый адрес устройства и идентификатор события, соответствующий его прерыванию. Если параметр access_flag не равен 0, устанавливается флаг занятости в дескрипторе. Процедура возвращает либо идентификатор устройства >0, либо код ошибки < 0;

г) ошибки:

```
    - 1            - устройство занято,
    - 2            - таблица переполнена,
    - 3            - модуль дескриптора в памяти не
                   обнаружен;
```


4) UnlinkDev: окончание работы с устройством:

а) обращение:

```
    void UnlinkDev(devID)
      unsigned short devID;
```

б) параметр:

```
    devID     (input)      - идентификатор устройства;
```

в) действие:

процедура удаляет устройство с заданным идентификатором из таблицы устройств трапхендлера и сбрасывает флаг занятости в дескрипторе;

г) ошибка :

```
    -6             - устройство не зарегистрировано для данной
                   задачи;
```

5) Readr8/Readr16/Readr32: чтение регистра:

а) обращение :

```
    char Readr8(regID)
      unsigned regID;
    unsigned short Readr16(regID)
      unsigned regID;
    unsigned Readr32(regID)
      unsigned regID;
```

б) параметр :

```
    regID     (input)      - идентификатор регистра;
```

в) действие:

производится чтение соответственно 8/16/32-разрядного регистра, значение которого выдается в качестве значения процедуры;

г) замечание:

в случае ошибки, а именно если заданный идентификатор регистра относится к незарегистрированному устройству, операция не выполняется;

6) Writer8/Writer16/Writer32: запись в регистр:

а) обращение:

```
    Writer8(regID, value)
      unsigned regID;
      char value;
    Writer16(regID, value)
      unsigned regID;
      unsigned short value;
```

```
Writer32(regID, value)
  unsigned regID, value;
```

б) параметры:

regID	(input)	- идентификатор регистра,
value	(input)	- новое значение регистра;

в) действие:

производится запись значения value в соответственно 8/16/32-разрядный регистр;

г) замечание:

в случае ошибки, а именно если заданный идентификатор регистра относится к незарегистрированному устройству, операция не выполняется;

7) Bitsr8/Bitcr8: сброс/установка бита в регистре:

а) обращение:

```
unsigned Bitsr8(regID, nbit)
  unsigned regID, nbit;
unsigned Bitcr8(regID,nbit)
  unsigned regID,nbit;
```

б) параметры:

regID	(input)	- идентификатор регистра,
nbit	(input)	- номер разряда;

в) действие:

производится установка/сброс бита nbit в регистре regID. Прежнее значение бита (0 или 1) возвращается в виде значения процедуры;

г) замечание:

в случае ошибки, а именно если заданный идентификатор регистра относится к незарегистрированному устройству, операция не выполняется;

8) Read_b8/Read_b16: групповое чтение:

а) обращение:

```
Read_b8(reg_list, data_list, n)
  unsigned reg_list[];
  char *data_list[];
  int     n;
Read_b16(reg_list, data_list, n)
  unsigned reg_list[];
  unsigned short *data_list[];
```



```
  int     n;
```

б) параметры:

reg_list	(input)	- список идентификаторов регистров;
data_list	(output)	- список указателей переменных;
n	(input)	- длина списков;

в) действие:

поочередно в соответствии со списком reg_list выполняется чтение 8/16-регистров, значение которых помещается по адресам, указанным в списке data_list. В качестве устройства для всех этих операций принимается устройство, к которому относится первый регистр из списка. Перед чтением первого регистра запрещаются прерывания, соответствующие уровню прерываний устройства. После чтения последнего регистра разрешение прерываний восстанавливается;

г) замечание

в случае ошибки, а именно, если заданный идентификатор регистра относится к незарегистрированному устройству, операция не выполняется;

9) Write_b8/Write_b16: групповая запись:

а) обращение:

```
Write_b8(reg_list, data_list, n)
  unsigned reg_list[];
  char *data_list[];
  int     n;
Write_b16(reg_list, data_list, n)
  unsigned reg_list[];
  unsigned short *data_list[];
  int     n;
```

б) параметры:

reg_list	(input)	- список идентификаторов регистров;
data_list	(input)	- список указателей значений;
n	(input)	- длина списков;

в) действие:

поочередно в соответствии со списком reg_list выполняется запись 8/16-регистров значений, указанных в списке data_list. В качестве устройства для всех этих операций принимается устройство, к которому относится первый регистр из списка. Перед чтением первого регистра запрещаются прерывания, соответствующие уровню прерываний устройства. После чтения последнего регистра

разрешение прерываний восстанавливается;

г) замечание :

в случае ошибки, а именно если заданный идентификатор регистра относится к незарегистрированному устройству, операция не выполняется.

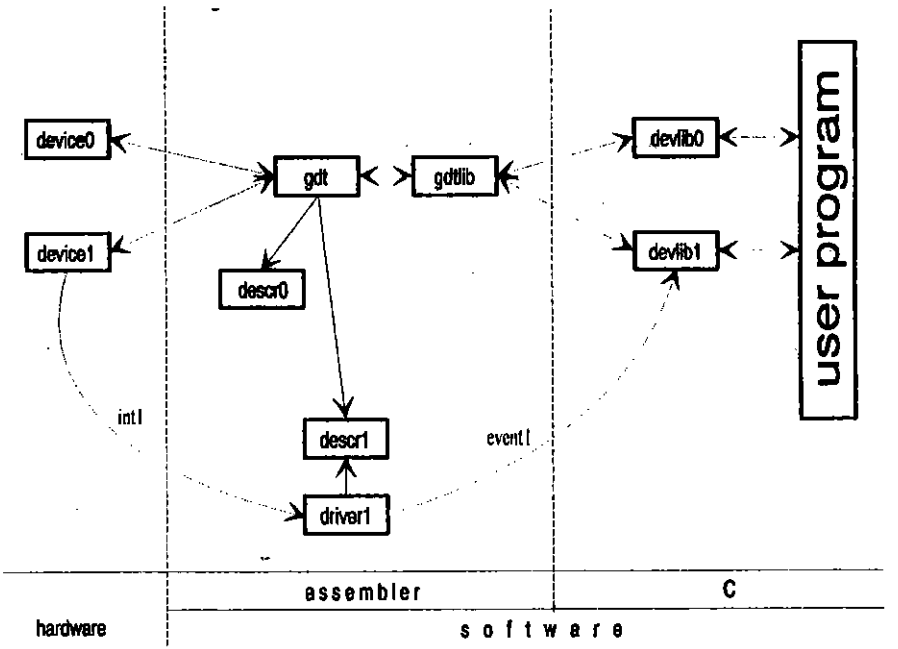

Схема взаимодействия программы пользователя с устройствами dev0 и dev1 при помощи gdt и gdtlib.

Рукопись поступила в издательский отдел
26 апреля 1994 года.