УДК 520.272.5

ЛОКАЛЬНЫЙ КОМПЛЕКС ДЛЯ СБОРА ДАННЫХ И УПРАВЛЕНИЯ РАДИОМЕТРАМИ С ПОДДЕРЖКОЙ РЕЖИМА ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

Б. Л. Ерухимов, О. П. Лихван,
А. В. Чепурнов, В. Н. Черненков

Рассмотрена реализация системы сбора данных и управления радиометрами, использующая механизм параллельных процессов. Система представляет собой развитие нижнего уровня описанной ранее многопользовательской системы сбора и предварительной обработки данных, базирующейся на двухуровневом многомашинном комплексе РАТАН-600.

The system for data acquisition and control of radiometers, which uses the mechanism of parallel processes, is considered. This system is the development of the subsystem of the earlier described complex for data acquisition and preliminary processing. The latter is based on the two-level multimachine complex of the RATAN-600 radiotelescope.

Последовательная реализация потенциальных возможностей радиотелескопа РАТАН-600 приводит к необходимости обеспечения прохождения постоянно возрастающего потока данных от радиотелескопа к архиву наблюдателя. Решение этой проблемы затруднено ввиду существенного и, более того, болезненного отставания доступных средств вычислительной техники и автоматизации научных исследований от требуемого уровня (в первую очередь, по мощности и надежности). Одним из путей преодоления подобных трудностей является максимальная оптимизация использования этих средств для построения систем сбора данных, а также выбор режимов работы, в которых эти средства все-таки могут обеспечить приемлемую надежность. Естественно, что резервы этого пути ограничены, что особенно ощущается в настоящее время. Тем не менее результаты работы авторов в этом направлении позволили в несколько раз повысить эффективность радиотелескопа. Приводимая ниже реализация локального комплекса является результатом работы, выполненной в том же русле, а именно: комплекс реализован с помощью аппарата параллельных процессов, поддержка которых не является естественной для систем на базе Э-60, ДВК-3, не имеющих механизма защиты памяти.

Поддержка параллельных процессов в той или иной мере является практически обязательным атрибутом стандартных операционных систем, с помощью которых строятся современные комплексы автоматизации научных исследований. Аппарат параллельных процессов широко используется при разработке диалоговых интерактивных пакетов, позволяющих исследователю одновременно контролировать несколько физических механизмов (сбор данных, предварительная обработка,

контроль аппаратуры, хронометраж), меняя при необходимости соответствующие параметры эксперимента. Технология проектирования программных продуктов, учитывающая возможности запуска параллельных процессов, в настоящее время развита в значительной степени. При этом проблема разработки новых базовых средств поддержки параллельных процессов (планировщиков) не теряет актуальности. Последнее также связано с частой необходимостью реализации планировщика на конкретных вычислительных структурах, непосредственно входящих в состав экспериментальной установки.

В системах автоматизации процессов наблюдений и инструментальных исследований, применяемых в САО АН СССР на телескопах РАТАН-600 и БТА, достаточно типовым является использование структур, описанных Ерухимовым и Черненковым (1987), Витковским и др. (1988). Эти структуры представляют собой радиальные многомашинные комплексы с хост-ЭВМ в центре и с сателлитными бездисковыми микро-ЭВМ в ветвях. При этом отсутствие дисковых операционных сред на сателлитных ЭВМ не является обязательным, их обычно просто не используют при разработке конкретных прикладных систем ввиду, как было сказано выше, малой надежности доступных дисковых накопителей. В математическом обеспечении таких комплексов можно выделить три уровня, расположенных по степени "приближения" к пользователю:

- стандартная многопользовательская операционная система, поддерживающая независимую работу нескольких терминалов хост-ЭВМ, в качестве которых могут выступать и сателлитные ЭВМ, работающие в режиме эмуляции терминала;
- инструментальная система поддержки радиальной сети сателлитных ЭВМ, обеспечивающая разработку и отладку программ для них, телезагрузку рабочего модуля и полный протокол файлового обмена между программой в сателлитной ЭВМ и внешними устройствами хост-ЭВМ (Ерухимов и Черненков, 1987; Ерухимов и Черненков, 1988);
- прикладная система с развитым пользовательским интерфейсом, обеспечивающая "прозрачное" взаимодействие наблюдателя или инженера с необходимыми аппаратно-программными слоями комплекса.

В данной работе речь идет о развитии второго из указанных уровней. Описывается модуль планировщика, включение которого в инструментальную систему сателлитной ЭВМ позволяет разрабатывать для нее конкретные программы, реализующие режим разделения времени между отдельными процессами. Обоснованием включения режима разделения времени в обеспечение локального комплекса является возможность существенного повышения сервиса наблюдателя. Так, с помощью этого режима наблюдатель, не нарушая стандартного процесса регистрации, может уточнять параметры наблюдения, выполнять сервисные процедуры, выбирать удобный режим визуализации регистрируемых данных и т.д. Крайне полезным этот режим оказывается также и для построения систем инженерного контроля радиометров.

Отличие предлагаемого планировщика от известных для отечественных микро-ЭВМ реализаций (например, Корнилов и Костин, 1986) заключается в следующем:

- сателлитная микро-ЭВМ может работать в режиме "STAND ALONE", т.е. лишена стандартной дисковой операционной среды;
- сателлитная микро-ЭВМ работает в синхронном режиме регистрации и необходимой обработки внешней информации, поступающей по прерываниям, т. е. в реальном времени, что ограничивает работу планировщика и поддерживаемых им

процессов оставшейся частью процессорного времени.

Рассматриваемый планировщик поддерживает программы, скомпилированные с языков СИ, ФОРТРАН и АССЕМБЛЕР в средах, подобных RT-11, при этом головной модуль должен быть написан на языке СИ. Собственно модуль планировщика представляет собой простой ассемблерный текст, в котором использована макробиблиотека структурных расширений MACRO 11/SP. Модули, запускаемые в качестве процессов, оформляются как СИ-функции, для запуска в качестве процессов фортрановских или ассемблерных модулей используется стандартная для языка СИ функция (call - интерфейс).

Работа программы в сателлитной ЭВМ, использующей планировщик, происходит следующим образом. Пусть программа находится в режиме выполнения какого-либо процесса, обычно фонового по сравнению с процедурами реального времени. Подобной процедурой реального времени является, например, регистрация данных, привязанных к жесткой координатно-временной сетке, с минимальным их преобразованием и последующей записью в буфер хранения. Такая процедура запускается по таймерному прерыванию, синхронизирующему работу машины. После этого управление передается планировщику. Результатом работы планировщика является планирование запуска текущего или следующего по очереди процесса, после чего осуществляется возврат из прерывания к выполнению запланированного процесса.

Идентификация процесса, который должен быть запланирован для запуска в данный квант времени, производится в соответствии с его местом в кольцевой очереди и приоритетом. Текущий приоритет процесса определяется переменной execnt, величина которой соответствует числу тактов времени, оставшихся для выполнения данного процесса до замены его другим. Например, если execnt = 5, планировщик будет запускать текущий процесс в течение 5 тактов, уменьшая каждый раз execnt на 1. При уменьшении execnt до 0 происходит переключение на планирование следующего по очереди процесса. Существует возможность запуска процесса с максимальным приоритетом, для чего используется флаговая переменная prtas.

Работа планировщика поясняется блок-схемой на рис.1. Для каждого процесса в соответствии с общепринятой идеологией (Кейслер, 1968; Корнилов и Костин, 1986) в памяти резервируется область, состоящая из сервисного блока и стека. На первом шаге планировщика проверяется флаг prtas - "запрещение планирования". Если он не равен нулю, то работа планировщика вырождается и сводится к выходу из прерывания, управление передается ранее прерванному процессу с максимальным приоритетом. В случае prtas = 0 и ненулевого значения execnt последнее уменьшается на единицу и осуществляется выход из прерывания с передачей управления текущему процессу. При этом оставшееся время резидентности текущего процесса составляет execnt тактов. При достижении execnt=0 выполняется основная функция планировщика - переключение процессов. В сервисный блок записывается срез текущего процесса - состояние регистров и дополнительная информация, например, состояние контроллера КАМАК. Далее из очереди выбирается следующий процесс, из сервисного блока восстанавливается состояние его регистров и приоритета, устанавливаются соответствующие значения prtas и execnt и осуществляется выход из прерывания с передачей управления следующему процессу.

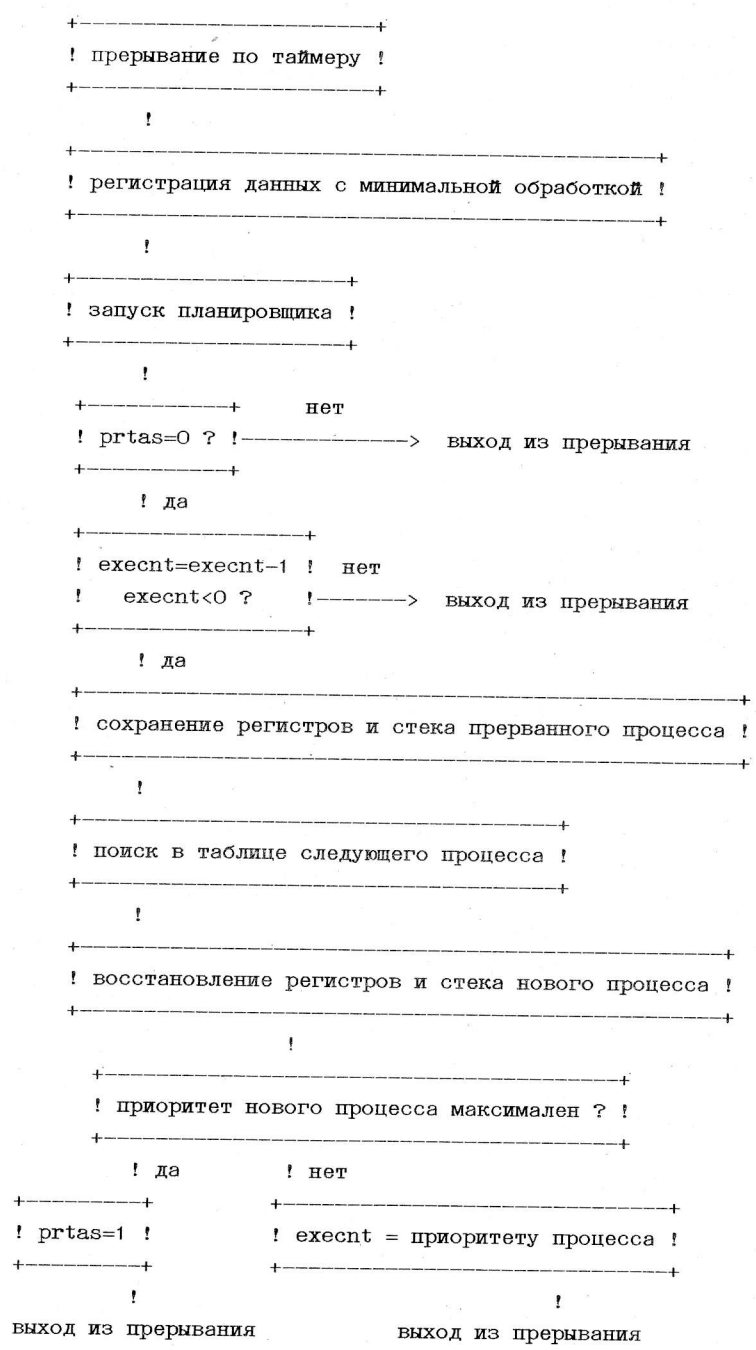

Рис. 1.

Ниже приводятся примитивы планировщика, реализованные в виде СИ-функций. Среди них можно выделить два основных вызова – это "инициализация таблицы процессов" и "запуск функции как процесса". Остальные примитивы предназначены

для реализации дополнительных возможностей управления процессами, таких как остановка и временная приостановка процесса, изменение приоритета процесса, возобновление работы процесса.

Тексты модулей, реализующих примитивы планировщика, вместе с базовыми константами находятся в специальном файле. Перед компиляцией файла нужно указать требуемые значения этих констант. Ниже приведен конкретный пример области указанного файла, содержащей редактируемые константы.

```
#define STACKSIZE 300  /* размер стека для одного процесса */
#define NPRS 2  /* Число планируемых процессов */
#define REGNUM 7 /* Число сохраняемых ячеек */
#define PRTYMAIN 5 /* Приоритет функции main() */
```

В этом примере во время работы можно одновременно запланировать два процесса, не учитывая основной программы, приоритет которой равен 5. Размер стека, выделяемого для каждого процесса, составляет 300 слов. Константу REGNUM=7 изменять не рекомендуется, это число сохраняемых регистров, если же требуется дополнительно сохранить какие-либо параметры (состояние контроллера КАМАК), необходимо внести очевидные поправки в ассемблерный текст планировщика MNGR.MAC, который не приводится в данной работе.

Перед запуском процессов обязательно необходимо вызвать (только 1 раз) функцию initial(). Примитив планирования запуска пользовательской СИ-функции name как процесса выглядит следующим образом:

```
on(name,arg,prty).
```

Приоритет процесса (prty) – любое целое число больше 0, определяющее количество временных тактов резидентности процесса; по истечении такого количества прерываний планировщик заменит старый процесс новым. Значение prty=0 означает запрещение планирования других процессов, т.е. процесс с таким (максимальным) приоритетом будет выполняться без разделения времени с другими процессами до полного своего завершения. Параметр arg позволяет передать любое значение типа int или адрес блока параметров запускаемой функции. Если параметр arg отсутствует, то достаточно обращения on(name, 0, prty).

Примитивы планировщика, управляющие процессами:

```
on(name,arg,prty)
setprf(name,prty) /* изменить приоритет */
off(name,flag)    /* прекратить или приостановить процесс */
prsstp()          /* немедленное прекращение процесса */
resetf(name)      /* вывод из останова */
prtas=1           /* запретить работу планировщика */
prtas=0           /* разрешить работу планировщика */
execnt=prty       /* временное изменение приоритета */
  int (*name)();  /* имя функции */
  int arg;        /* аргумент, передаваемый функции name */
  int prty;       /* приоритет */
  shar flag;      /* 0 или 1 */
  int prtas,execnt;
```

Обращение off(name,0) означает остановку процесса с именем name без освобождения его стека, работа процесса может быть продолжена после вызова resetf(name). Обращение off(name,1) приведет к уничтожению процесса с освобождением стека и места в таблице, которые могут быть использованы при запуске нового процесса. Все примитивы, кроме initial() и prsstp(), при успешном завершении возвращают 1, в противном случае – 0. Чтобы убедиться в активности процесса, можно воспользоваться вызовом setprf(name,prty).

Пример:

```
main()
{
   static int buff[3]={1,2,3};
   int func1(),func2(),buff[100];
initial();
on(func1,0,20);
on(func2,buff,4);
for(;;) puts("work main");
}
func1()
{ for(;;) puts("work func1"); }
func2(a)
int *a;
{ register int i;
printf("%d %d %d",*a++,*a++,*a);
for(i=1;i<=20000;i++;) puts("work func2");
}
```

В этой программе вывод на терминал поочередно будут вести три процесса – main, func1 и func2. Через некоторое время процесс func2 завершится, оставшиеся два процесса будут делить время между собой. Следует отметить, что функция main в качестве процесса ничем не отличается от других процессов, и ею можно так же управлять с помощью перечисленных примитивов.

На основе рассмотренного планировщика реализована последняя версия программы регистрации данных и контроля радиометров на облучателе N 1 РАТАН-600. Как было сказано выше, эта программа представляет собой одно из звеньев более крупной системы поддержки наблюдений на радиометрах сплошного спектра РАТАН-600, последняя версия которой описана в работе (Витковский и др. 1989). С помощью этой системы предварительно на хост-ЭВМ готовится пакет заданий на наблюдения некоторого количества источников в соответствии с их временами кульминации. Далее в локальный комплекс сбора загружается рабочая программа, которая в соответствии с пакетом заданий организует наблюдения соответствующих источников, периодически сбрасывая накопленные данные в соответствующие файлы на диске хост-ЭВМ. Наблюдения проводятся в автоматическом режиме и, в принципе, не требуют участия наблюдателя. Однако использование планировщика позволяет реализовать интерактивный режим на фоне автоматического, что существенно повышает уровень сервиса наблюдателя.

В программе используются три параллельно работающих процесса, модифицирующие свой приоритет при соответствующих условиях. Главный процесс (main) выполняет все основные функции по регистрации данных, управлению

радиометрами и записи данных на диск хост-ЭВМ, а также отображению звездного времени. Кроме этого,он запускает процесс визуализации, который выводит данные на видеоконтрольное устройство (телевизор) квазисинхронно с регистрацией, а также процесс "help", представляющий собой монитор наблюдателя. После запуска "help" сообщает о том, что он активен и с пониженным приоритетом ждет ввода с терминала символа "H". После ввода этого символа процесс "help" повышает свой приоритет и печатает на дисплее меню:

'I' - > ВВОД КОММЕНТАРИЯ
'T' - > ТЕСТ СВЯЗИ С ХОСТ-ЭВМ
'A' - > РАСЧЕТ УРОВНЯ ШУМА
'G' - > ПРЕКРАЩЕНИЕ ТЕКУЩЕГО НАБЛЮДЕНИЯ С СОХРАНЕНИЕМ ДАННЫХ
'E' - > ИНФОРМАЦИЯ О ПАРАМЕТРАХ ТЕКУЩЕГО НАБЛЮДЕНИЯ
'C' - > ПЕРЕХОД К СЕРВИСНОЙ ИНЖЕНЕРНОЙ ПРОГРАММЕ

Меню включает в себя процедуры, которые можно активизировать вводом с терминала соответствующего символа и в итоге получить нужную информацию или выполнить необходимые действия. По команде 'A', например, на фоне процесса регистрации и процесса визуализации данных на экране видеоконтрольного устройства, а также отображения текущего звездного времени осуществляется расчет уровня "шум - дорожки" радиометра с использованием помехоустойчивого алгоритма абсолютного медианного отклонения. Аналогичным образом строится сервисная инженерная программа, где преимущества режима параллельных процессов еще более очевидны. Ниже представлен вариант меню этой программы:

'H' - > ВЫВОД НА ТЕРМИНАЛ СПИСКА КОМАНД
'L' - > ТЕСТ СВЯЗИ С ХОСТ-ЭВМ
'U' - > УСТАНОВКА УРОВНЯ УСИЛЕНИЯ
'M' - > РУЧНАЯ КОМПЕНСАЦИЯ
'A' - > АВТОМАТИЧЕСКАЯ КОМПЕНСАЦИЯ
'T' - > РАСЧЕТ ПОСТОЯННОЙ ВРЕМЕНИ И УРОВНЯ ШУМА КАНАЛА
'V' - > ВИЗУАЛИЗАЦИЯ ВСЕХ КАНАЛОВ
'C' - > ВКЛЮЧЕНИЕ КАЛИБРОВКИ ПО ВСЕМ КАНАЛАМ

Некоторые процедуры, например 'U', 'M', 'T',имеют внутренний уровень диалога, где пользователю предоставляется соответствующее меню. В целом,использование планировщика существенно повышает модульность программы и позволяет легко ее адаптировать при модернизации аппаратуры.

Авторы выражают признательность В. С. Шергину за полезные консультации в процессе разработки.

ЛИТЕРАТУРА

Витковский В.В., Ерухимов Б. Л., Малькова Г. А., Черненков В. Н.: 1988, Сообщ. Спец. астрофиз. обсерв., 58, С. 5-17.

Витковский В. В., Ерухимов Б.Л., Малькова Г.А., Мингалиев М.Г.,Черненков В.Н.: 1989, Препр. САО АН СССР, No. 28, 24 с.

Ерухимов Б. Л., Черненков В. Н.: 1987, Астрофиз. исслед. (Изв. САО), 24, С. 183–190.

Ерухимов Б.Л., Черненков В.Н.: 1988, Препр. САО АН СССР, 17, 10 с.

Кейслер С.: 1986, Проектирование операционных систем для малых ЭВМ. Пер. с англ. под редакцией С.А.Усова. М.: Мир, 680 с.

Корнилов А.Р., Костин А.Е.: 1986, Планировщик параллельных процессов для ОС ДВК. Микропроцессорные средства и системы, 4, С.56–58.

Поступила в редакцию
9 декабря 1989 г.