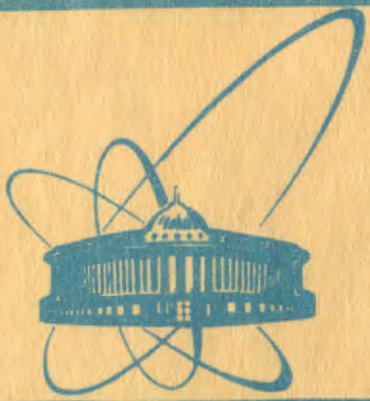


сообщения
объединенного
института
ядерных
исследований
дубна

11-81-488

И.И.Шелонцев, Н.Ю.Ширикова

О НЕКОТОРЫХ ВОЗМОЖНОСТЯХ ОПТИМАЛЬНОГО ИСПОЛЬЗОВАНИЯ ЭВМ CDC-6500

1981

Настоящая работа посвящена вопросам оптимального использования транслятора FTN /1/, отладки программ, экономии бумаги и времени, затрачиваемых при их перетрансляциях.

Коснемся вопроса отладки. Статистика показывает, что на отладку программы и передачу на счет тратится 60-75% времени, отведенного на ее создание. Однако проверка работы CDC-6500 за сутки показала, что практически во всех задачах, оставленных на ночной счет или запущенных с терминала, была трансляция программ (20% с ошибками). Настораживает и то обстоятельство, что многие пользователи вносят изменения в программу и не предусматривают специальной их отладки. Легкость исправления формулы на алгоритмическом языке высокого уровня является только кажущейся. Можно привести много примеров, когда одно "безобидное" исправление программы, такое, как вставленная печать, прекращало счет. Отладка программы - дело кропотливое, требующее от программиста большого искусства в придумывании различных вариантов данных для проверки правильности ее работы. И поэтому следует использовать возможности, предоставляемые операционной системой для отладки. Транслятор FTN предоставляет несколько таких возможностей. Во-первых, существует специальный режим D , предназначенный для этой цели. В настоящей работе мы не будем рассматривать этот режим, а коснемся других, дополнительных возможностей. Транслятор FTN имеет параметры A,ER,T, которые следует использовать при отладке программы.

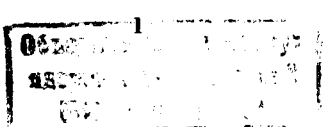
A - параметр выхода.

Если во время трансляции были обнаружены фатальные ошибки, то операционная система начинает выполнять управляющие карты задачи (если такие есть), стоящие после карты EXIT(S).

Если карты EXIT(S) нет, то счет задачи прекращается.

При работе в интерактивном режиме параметр A не применим.

Этот параметр необходимо использовать, если в задаче, например после трансляции, предусмотрено создание библиотеки. Если в под-



программах, включаемых в библиотеку, были при трансляции обнаружены фатальные ошибки, то попытка создания библиотеки будет бесполезной тратой времени.

ER Если при вызове транслятора указать этот параметр, то транслятор специально вставит в программу коды, которые при счете определят причину прерывания счета. Распознаются следующие ошибки: арифметические действия с неопределенными и бесконечными числами; нераспознаваемый запрос задачи к операционной системе, посылаемый в ячейку RA+1 (первую ячейку поля); исчерпание заказанного времени счета центральным процессором или операциями ввода-вывода; попытка записать на диск больше, чем он может вместить . Фиксируется также прекращение счета задачи оператором. Если ошибка произошла, когда выполнялась команда, находящаяся в поле задачи, то в протокол прохождения задачи выдается имя подпрограммы и номер строки текста, где произошла ошибка. Следует учитывать, что если транслятор был вызван с режимом OPT=I или OPT=2, то номер строки может быть приблизительным, поскольку оптимизация может привести к изменению порядка выполнения операторов. Если ошибка произошла при попытке выполнить команду (записать или прочитать числа, расположенные вне поля задачи), то выдается только содержимое P-регистра (адрес выполняемой команды).
Режим ER увеличивает длину программы и должен быть использован только при отладке программ.

Если транслятор FTN вызван в режиме TS , OPT=0, то режим ER задается автоматически.

T – параметр прослеживания ошибки.
Производится полное прослеживание ошибки (имя подпрограммы, где была обнаружена ошибка, какая подпрограмма к этой подпрограмме обращалась и т.д.). Этот режим также предназначен для отладки программы. Вызов транслятора FTN с режимом D (отладки) или OPT=0 автоматически задает режим T. Список ошибок, которые могут быть прослежены, указаны в приложении В руководства по транслятору FTN .

Ниже приводится диагностика, выдаваемая транслятором FTN , если он был вызван с режимом ER (строки отмечены чертой):


```
MODE ERROR
JOB REPRIEVED
  INFINITE VALUE IN TEST NEAR LINE 12
     . . . CP SECOND EXECUTION TIME
(PREVIOUS ERROR CONDITION RESET)
ERROR MODE = 02  ADDRESS = 002240
  .
  .
  .
```

Если параметр T не указан при обращении к FTN , то, например при извлечении квадратного корня из отрицательного числа, будет выдана следующая диагностика:

```
ARGUMENT NEGATIVE                    (отрицательный аргумент)
ERROR NUMBER 39 DETECTED BY SQRT
(ошибка номер 39 обнаружена программой SQRT )
```

В таком случае, чтобы установить, какое же именно обращение к SQRT дало ошибку, придется просмотреть все обращения. Добавление параметра T обеспечит следующую диагностику (строки отмечены чертой):

```
ARGUMENT NEGATIVE
ERROR NUMBER 39 DETECTED BY SQRT
CALLED FROM E AT LINE 3
CALLED FROM TEST AT LINE 12
```

Транслятор FTN предоставляет возможность выдачи таблицы ссылок, в которой в алфавитном порядке указываются имена всех переменных программы, дается список всех меток , всех подпрограмм, вызываемых данной подпрограммой, и многое другое. Таблица ссылок выдается в трех видах при помощи параметра R =I,2,3. Самый полный вид ссылок получается при задании режима R =3. В этом случае для каждой переменной указывается, в каких строках программы переменная используется и где она определена. Этот режим следует применять только в двух случаях: при выдаче текста чужой программы, которую предстоит запустить в эксплуатацию, и при подготовке документации для уже отлаженной программы. При отладке программы, как правило, достаточно использовать таблицу ссылок, выдаваемую режимом R= I. Особое внимание при анализе таблицы следует обращать на те переменные, которые транслятором опознаны как неопределенные - *UNDEF . Практика показывает, что внимательный просмотр таблицы позволяет заметить неправильные пробивки, забытые описания массивов и многое другое.

Рассмотрим более подробно таблицу ссылок, выдаваемую в режиме R = I.

Точки входа (ENTRY POINTS)

Имена, включенные в эту таблицу, являются именами программы (подпрограммы) и именами из операторов ENTRY .

Формат таблицы следующий:

ENTRY POINTS		DEFINITION	REFERENCES
относ. адрес	имя	номер строки (нет для R= I)	для подпрограммы номер строки оператора RETURN (нет для R = I)

Пример:

```
ENTRY POINTS
4011 MAPS
```

Переменные (VARIABLES)

В эту таблицу включаются имена локальных переменных, переменных из общих блоков, массивов, формальных параметров, имена, присвоенные операторам RETURNS , а для подпрограмм-функций включается имя подпрограммы. Формат таблицы следующий:

VARIABLES		SN	TYPE	RELOCATION
относ. адрес	имя	* (нет для R =I)	тип	свойство блок ссылки

Имена приводятся в алфавитном порядке.

* SN=stray name flag. (Указание, что переменная появилась только раз в программе, чтобы обратить внимание на возможную ошибку в имени переменной).

ТИП LOGICAL, INTEGER, REAL, COMPLEX или DOUBLE
RETURNS , если имя есть формальный параметр в операторе RETURNS.

свойство
- *UNDEF Переменная не была определена. (Она не появилась в COMMON- или DATA -операторах, в левой части оператора присваивания, не является переменной цикла, не появилась в виде фактического параметра при обращении к подпрограмме-функции или подпрограмме, не появилась в списке ввода).
- ARRAY Если переменная с индексом.
- *UNUSED Неиспользуемый формальный параметр.

блок имя COMMON блока, где указано имя переменной.
/ / - для переменной из непомеченного блока.

F.P. - для указания, что эта переменная является формальным параметром.
пробел - для локальной переменной.

ссылки
- REFS и номера строк, где эта переменная использовалась.
- DEFINED и номера строк, где переменной присваивается значение.
- IO REFS и номера строк, где переменная была использована в качестве имени файла в операторах ввода, вывода.

Пример таблицы для R = I:

VARIABLES	SN	TYPE	RELOCATION				
4176	SIZE1	INTEGER		4177	SIZE2	INTEGER	
4175	STRAY	INTEGER	*UNDEF	4170	S1	INTEGER	

Имена файлов (FILE NAMES)

В эту таблицу включаются имена файлов, которые были указаны в операторе PROGRAM или были неявно определены в подпрограммах, в операторах ввода/вывода. Формат таблицы следующий:

FILE NAMES		MODE	
относ. адрес	имя	режим записи/ чтения	ссылка (нет для R = I)

адрес Адрес таблицы FIT файла, буфер начинается с адреса+34$_8$. В подпрограмме вместо адреса стоит пробел.

имя Имя файла, определенное в операторе PROGRAM , или имя, полученное из оператора ввода/вывода. Например, оператор WRITE(2) в подпрограмме подразумевает запись в файл TAPE2.

режим записи/чтения
- FMT Форматный, например, READ(2,901)
- FREE Свободный, например, READ(2,*)
- UNFMT Бесформатный, например, READ(2,*)
- NAME Поименный, например, READ(2,NAMEIN)
- BUF Буферный, например , BUFFER IN(2,0)
- MIXED Смешанный (несколько комбинаций, указанных выше).
- Пробел Режим не определен.

ссылки
- READS и номера строк, где были операторы чтения этого файла.
- WRITES и номера строк, где были операторы записи в этот файл.

MOTION и номера строк, где встречались операторы, двигающие указатель положения на файле, например, REWIND, BACKSPACE, ENDFILE .

Если переменная используется как номер устройства в операторах ввода /вывода, то печатается:

VARIABLE USED AS FILE NAME, SEE ABOVE .

Пример таблицы:

```
R=1:
    FILE NAMES MODE
     0   INPUT        2041  OUTPUT FMT  0  TAPE5  NAME
R=2,  R=3  :
    FILE NAMES MODE
     0   INPUT
    2041 OUTPUT FMT  WRITES  9  12
     0   TAPE5  NAME READS   7
```

Внешние ссылки (EXTERNAL REFERENCES)

В таблицу внешних ссылок включаются имена функций или подпрограмм, явно вызываемых в программе или подпрограмме, а также имена, указанные в операторах EXTERNAL.

Формат таблицы следующий:

EXTERNALS имя	TYPE тип	ARGS кол-во аргументов	свойство	REFERENCES ссылки (нет для R =1)

тип Тип функции.
Пробел, если имя было именем подпрограммы.

свойство F.P. - имя было формальным параметром.
LIBRARY-имя было именем библиотечной программы, вызываемой по значению.

Пример таблицы:

```
R=1:
    EXTERNALS    TYPE   ARGS
      NOHEAD              1        PASCAL       1

R=2, R=3:
    EXTERNALS    TYPE   ARGS       REFERENCES
     NOXEAD              1            14
     PASCAL              1            11
```


Таблицы внешних ссылок следует смотреть особо внимательно. Так, например, оператор

A=W(I) ,

если было пропущено описание DIMENSION W(...), будет интерпретирован как обращение к внешней подпрограмме w с одним аргументом.

Встроенные функции (INLINE FUNCTIONS)

В таблицу встроенных функций включаются имена библиотечных функций и операторов-функций, которые встречаются в программе.

Формат таблицы следующий:

INLINE FUNCTIONS имя	TYPE тип	ARGS кол-во арг.	DEF функ. тип	LINE номер строки определения	REFERENCES ссылки

тип Арифметический тип.
NO TYPE , если в выражении присутствуют величины разных типов.

количество аргументов Количество аргументов, которые указаны при обращении к функции. Для функций с переменным количеством аргументов (таких, как MAX,AND и др.) проставляется 0.

функциональный тип
INTRIN - для библиотечной функции.
SN - для функции-оператора.

Пример таблицы:

```
R=1:
INLINE FUNCTION TYPE ARGS
    MAXO      INTEGER  0  INTRIN  MINO  INTEGER  0  INTRIN

R=2,R=3:
INLINE FUNCTION TYPE ARGS   DEF  LINE  REFERENCES
    MAXD      INTEGER  0   INTRIN          8
    MINO      INTEGER  0   INTRIN         10
```

Затем следует таблица объединенных групп ввода/вывода, если такие операторы есть в программе (NAMELIST,...)

Таблица меток (STATEMENT LABELS)

Эта таблица включает все метки, определенные в программе или подпрограмме. Метки расположены в таблице по возрастающим номерам.

Формат таблицы следующий:

STATEMENT	LABELS		DEF LINE		REFERENCES
относ. адрес	метка	тип	активность	номер строки определения	ссылки

относительный адрес — Присвоенный относительный адрес.
0 - для неактивной метки и для метки, которая является меткой последнего оператора цикла.
400000, если адрес не присвоен (обычно при ошибках в программе).

тип — FMT , если метка оператора FORMAT.

UNDEF , если метка не была определена. В ссылках указаны номера строк, где эта метка использовалась.

Пробел, если метка использовалась в правильных выполняемых операторах.

активность — INACTIVE Метка рассматривается неактивной и при оптимизации программы может быть вычеркнута транслятором. Относительный адрес неактивной метки - 0.

NO REFS Метка не используется ни в одном операторе. Ее можно убрать из программы.

Пробел. Если метка активная или на нее есть ссылка.

определение — Номер строки, где метка была определена.

Пример таблицы:

```
R=1:
STATEMENT  LABELS
   0          1      0   2   113     3     FMT
  76          4  FMT
R=2,  R=3:
STATEMENT     LABELS        DEF     LINE      REFERENCES
   0            1           13                   12
   0            2           14                    9
 113            3  FMT      15                   14
  76            4  FMT       5                    4
```

Таблица операторов цикла (DO LOOPS)

Эта таблица включает в себя все явные и неявные (в операторах DATA) циклы и указывает их свойства. Она не выдается при фатальных ошибках в программе или при трансляции в режиме 0.

Формат таблицы следующий:

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
относ. адрес начала цикла	метка последнего оператора цикла	имя переменной цикла	начальный последн. номера строк	длина в восьм. представл.	свойства

Для неявного цикла в качестве метки последнего оператора выдается пробел.

Если имя переменной цикла не используется в цикле, то это имя отмечается *.

свойства OPT — Цикл был оптимизирован.

INSTACK — Цикл занимает меньше 7 слов памяти.

EXT REFS — Цикл не оптимизирован, так как внутри него есть обращение к подпрограммам или он является неявным циклом в операторе ввода/вывода.

ENTRIES — Цикл не оптимизирован, так как в него есть входы извне.

NOT INNER — Цикл не оптимизирован, так как не является самым внутренним вложенным циклом.

EXITS — Цикл не оптимизирован, так как имеет ссылки на метки, расположенные вне его.

Пример таблицы:

```
R=1, R=2, R=3:
   LOOPS   LABEL   INDEX   FROM-TO   LENGTH    PROPERTIES
     20              *I        4        4B      EXT REFS
     43       2      *I        9 14     20B     EXT REFS NOT INNER
     52       1       J       12 13     2B      INSTACK
```

Выдаются еще таблица общих блоков, таблица эквивалентностей.

В конце каждой подпрограммы печатается статистика: длина программы (в восьмеричном и десятичном виде), длина буферов, общая длина помеченных общих блоков, непомеченных общих блоков.

Таким образом, при анализе таблицы ссылок особенно для программы, транслируемой в первый раз, следует обратить особое внимание на:
- имена, пробитые с ошибками;
- переменные, которые использовались только один раз;
- переменные, которые были опознаны как UNDEF;
- функции, которые должны быть массивами;
- функции, которые должны быть встроенными, а описаны как внешние;

- неправильный тип переменных или функций;
- неиспользуемые операторы формата;
- неиспользуемые формальные параметры;
- порядок переменных в общих блоках;
- эквивалентные величины.

Отметим, что транслятор FTN на самом деле представляет из себя набор нескольких вариантов транслятора, вызов которых осуществляется при помощи параметров Q,TS,OPT=0,OPT=1,OPT=2. Рассмотрим кратко возможности этих режимов.

Q Производится только самая быстрая трансляция. Для нее требуется мало слов оперативной памяти, примерно столько же, сколько и для режима OPT=0.
Этот режим следует использовать при трансляции программы в первый раз.

TS Быстрый режим трансляции. При трансляции требуется минимально 40000_8 слов памяти. Время счета программы примерно такое же, как и после трансляции в режиме OPT=0. Режим TS следует использовать в случае, если приходится часто перетранслировать программу, время счета которой не более чем в два раза больше времени трансляции. Рекомендуется использовать этот режим для трансляции небольших программ с терминалов.

Хотелось бы обратить особое внимание на использование параметра OPT. Транслятор FTN допускает три вида оптимизации. Не останавливаясь подробно на вопросах оптимизации, заметим, что

OPT=0 Режим быстрой трансляции, когда автоматически добавляются режимы ER и т. Этот режим следует использовать при отладке программ.

OPT=I Стандартная оптимизация. Минимальная память, необходимая для трансляции, - 46000_8 слов.

OPT=2 Режим максимальной оптимизации.
Очень медленный режим трансляции, для которой требуется от 54000_8 слов памяти и больше в зависимости от длины транслируемой программы. Этот режим следует использовать только для отлаженных задач. Обязательно следует сравнить результаты контрольного счета, выданные программой, транслируемой с режимом OPT=I и OPT=2, прежде чем начать массовый счет по программе.

Проиллюстрируем использование различных вариантов трансляции на примере программы оверлейной структуры, состоящей из 38 программ (2656 перфокарт)

режим	время трансляции	память при трансл.	время счета контр. варианта	память при счете	память для загрузки
Q	8.418	52000			
TS	29.638	45000	24.314	I00277	72500
OPT=0	25.I3I	52400	24.333	I0056I	73I00
OPT=I	26.086	52400	24.342	74750	67600
OPT=2	40.356	65500	24.383	74655	67400

На загрузку программы тратилось 4 секунды.

В настоящее время нет аппарата проверки программы на оптимальность. Однако подмечено, что пользователи, которые работали на машинах первых поколений с малой памятью и небольшим быстродействием, пишут программы более экономично. При написании программы, по которой будет вестись длительный счет, надо изучить недостатки транслятора, который будет использоваться для трансляции. Так, например, транслятор FTN не может построить оптимальную программу для вычисления таких выражений:

```
X=C*SIN(A)+D*COS(B)
Y=C*SIN(A)-D*COS(B)
```

Поэтому автору программы следует самому выносить вычисление функций от одного и того же аргумента в дополнительные операторы, например, так:

```
SINA=SIN(A)
COSB=COS(B)
X=C*SINA+D*COSB
Y=C*SINA-D*COSB
```

В таком варианте программы транслятор уже и сам поймет, что C*SINA и D*COSB нужно вычислять лишь один раз.

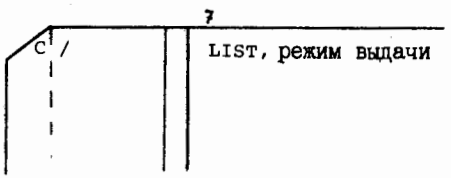
Нет смысла тратить время на убеждение, что бумагу следует экономить. Экономия бумаги - это закон, которому должны все подчиняться.

Анализ выдач показал, что программисты многих лабораторий не предусматривают в программах предварительный просмотр выдаваемого на печать. Можно увидеть гистограммы, в которых только одна или нет ни одной точки. Или, например, выдается на печать большой массив чисел, состоящих из одних нулей. Наверное, можно было бы предусмотреть выключение печати тех чисел, которые по модулю меньше какого-то заданного программистом числа.

Особенно много тратится бумаги при выдаче текста программ, которые перетранслируются по нескольку раз за день. Вряд ли в подпро-

граммах каждый раз происходит полное изменение. В апреле этого года в транслятор FTN было внесено следующее изменение: при печати очередной подпрограммы автоматического перехода на начало нового листа не происходит.

В трансляторе FTN предусмотрена возможность выключения печати части текста с помощью особых карт:



C/ должны быть пробиты в колонках 1,2, а в колонках 3-6 должны быть пробелы.

режим выдачи
- NONE Прекращается печать текста.
- ALL Возобновляется печать текста.

Пример: Программа транслировалась в режиме EL=A,R=3,OPT=1:
Эти карты вставляются в текст программы и производят действия, если при вызове транслятора предусмотрен режим выдачи текста (нет L=0). Действие этих карт распространяется только на ту программную единицу (программа, подпрограмма, функция), где они помещены. Обычно сами карты включения или выключения печати текста печатаются. Однако, если поставить LIST, NONE самой первой картой перед заголовком программы или подпрограммы, то печать данной программной единицы будет пропущена полностью (если не было фатальных ошибок). Если LIST, NONE стоит перед оператором END в подпрограмме, то таблица ссылок не выдается на печать. Если с помощью LIST,NONE была отключена печать части текста, в которой были обнаружены фатальные ошибки, то операторы, в которых есть ошибки, выдаются на печать.

```
      PROGRAM P
C/    COMMENT
C/    LIST,NONE
C/    COMMENT
      DIMENSION A(10)
      INTEGER B/C
C/    LIST,ALL
      DO 100  I=1,10
100   A(I)=0
C/    LIST,NONE
      RETURN
      END
```


На печать будет выдано:

```
1           PROGRAM P
     C/     COMMENT
3    C/     LIST,NONE
6           INTEGER B/C     - эта строка напечатана из-за наличия
                              ошибки.
7    C/     LIST,ALL
            DO 100  I=1,10
   100      A(I)=0 .
10   C/     LIST, NONE
```

Затем выдается только таблица ошибок и не выдается таблица ссылок.

Очень многие пользователи печатают расширенную таблицу загрузки при помощи MAP,ON. Для отладки программы достаточно выдавать таблицу загрузки при помощи MAP,PART . Режим расширенной выдачи нужен в основном лишь для изучения структуры чужой программы.

Как уже отмечалось, многие пользователи производят многочисленную перетрансляцию своих программ. Так, например, в январе 1981 г. на CDC-6500 были пропущены 16442 задачи, для которых транслятор FTN был вызван 18857 раз, редактор UPDATE - 8487 раз. Таким образом, практически в каждой задаче была трансляция текста, на что за месяц было потрачено 36 часов центрального процессора, т.е. в день 1,2 часа тратилось только на трансляцию. Вот пример типичной задачи, пропускаемой на CDC-6500:

```
.
.
.
ATTACH,OLDPL,...
UPDATE(F)
FTN(I)
LGO.
7/8/9
   коррекция библиотеки
7/8/9
.
.
.
```

В основном производится коррекция не всей библиотеки, а только ее части. Было бы более экономично использовать библиотеку в транслированном виде, заменяя в ней каждый раз откорректированную часть. Тем самым каждый раз потребуется меньшее время, необходимое для трансляции только части библиотеки. Системная программа COPYL предназначена для выполнения указанных действий. Она копирует подпрограм-

мы, записанные в транслированном виде на файле (назовем его основным файлом), на новый файл, подставляя те подпрограммы из файла замены, имена которых и тип (по желанию) совпадают с именами подпрограмм основного файла. Информация о работе COPYL выдается только в протокол задачи. Для COPYL требуется 12000_8 слов памяти.

Для приведенного примера более эффективно задачу пропускать следующим образом:

I этап	Последующие этапы
ATTACH,OLDPL,...	ATTACH,OLDPL,...
UPDATE(F)	UPDATE(Q)
REQUEST,LGO, *PF.	ATTACH,OLD,...
FTN(I)	FTN(I)
CATALOG,LGO,...	COPYL.
LGO.	NEW.
7/8/9	7/8/9
коррекция	новая коррекция
7/8/9	7/8/9
данные	данные
7/8/9	7/8/9

Это было проверено на библиотеке, состоящей из 28 подпрограмм, входящих в программу оверлейной структуры. Обычно пользователь тратил 46 с центрального процессора и 267 с периферийного процессора, из которых 13 с уходило на редактирование и 27 с на транслирование с использованием 52000_8 слов оперативной памяти. С использованием COPYL задача стала тратить 18 с центрального процессора и 79 с периферийного процессора, из которых на трансляцию было израсходовано 6 с (корректировались 4 подпрограммы). Таким образом, выигрыш только во времени был в 3 раза.

Обращение к программе COPYL:

COPYL(oldlfn,replfn,newlfn,last,flag)

oldlfn Имя основного файла.
Подразумеваемое имя - OLD.

replfn Имя файла замены.
Подразумеваемое имя - LGO.

newlfn Имя нового файла.
Подразумеваемое имя - NEW.

last Имя последней подпрограммы, которую нужно взять из основного файла. Если имя не указывается, то берутся все подпрограммы.

flag Режим работы. Задается с помощью комбинации перечисленных ниже букв. Если режим не указан при обращении, то программа работает без выборки режимов.

R Вернуть файлы oldlfn и newlfn в начальное положение перед переписью (replfn возвращается в начальное положение всегда).

A Добавить в конец нового созданного файла те подпрограммы из файла замены, имена которых не встретились в основном файле.

T При сравнении программ не проверять типы программ.

Эти режимы можно задавать, объединяя указанные буквы в любом порядке, например, TRA,AR,RT,TR.

Остановимся на наиболее часто встречающихся типах подпрограмм:

ABS Программы оверлейной структуры.

COS Программа была транслирована при помощи FTN или COMPASS, но при трансляции были обнаружены ошибки.

REL Транслированная программа.

При работе с программами типа COS COPYL обходится особым образом. Если программа типа COS появляется в файле замены, то она всегда игнорируется и в протокол задачи ничего не сообщается. Если программа типа COS имеется на основном файле и в файле замены есть программы с таким же именем, но другого типа, то программа из файла замены всегда переписывается на новый файл. Если же в файле замены такой же программы не встретится, то на новый файл программа переписывается с основного файла.

Информацию о том, какие программы и какого типа находятся на каком-нибудь файле, можно получить с помощью системной программы

ITEMIZE: ITEMIZE (lfn, p_1,p_2,...,p_n)

lfn Имя файла. Подразумеваемое имя - LGO. (Допускается иметь на этом файле несколько файлов).

L=listlfn Выдача о работе ITEMIZE записывается на файл listlfn. Стандартный подразумеваемый режим L=OUTPUT.

PD Плотность печати-8 строк на дюйм. Если режим не задан, то печатается по 6 строк на дюйм.

NR Не возвращать файл lfn в начальное положение перед работой и после работы ITEMIZE. Если NR не указан, то lfn устанавливается в начальное положение до и после работы ITEMIZE.

N=n	Просмотреть n (десятичное число) файлов, содержащихся в файле lfn. Стандартный подразумеваемый режим N=1.
N=O	Просмотреть, пока не встретится пустой файл.
E	Выдать дополнительную информацию о программе.

Имя файла должно быть первым параметром, порядок остальных параметров произвольный.

В стандартном режиме программа ITEMIZE выдает следующую информацию:

REC	Порядковый номер программы по файлу.
NAME	Имя программы.
TYPE	Тип программы.
LENGTH	Количество слов (восьмеричное) в программе.

Более подробно о программах COPYL и ITEMIZE можно прочитать в /2/.

Краткое замечание о работе загрузчика на CDC . Как и всякая программа, загрузчик также помещается в память машины. Для его работы может потребоваться до 70000 слов. Чтобы уменьшить общее количество слов памяти (загружаемая программа + загрузчик), можно применить следующий простой прием. Следует большой массив из программы, в который не надо предварительно при помощи оператора DATA засылать начальные значения, объявить непомеченным общим блоком, который и будет использован загрузчиком для своей работы.

Например, вместо оператора

```
DIMENSION  A(4000),B(4000)
```

использовать

```
COMMON//A(4000),B(4000)
```

В заключение авторы выражают благодарность В.П.Ширикову за проявленный интерес к работе и помощь, О.В.Благонравовой, Т.Н.Забой и Л.А.Калмыковой за помощь в анализе работы пользователей CDC-6500.

Литература

1. FORTRAN EXTENDED VERSION 4. REFERENCE MANUAL(60497800), Sunnyvale, California (USA).
2. CYBER COMMON UTILITIES REFERENCE MANUAL(60493300), Sunnyvale, California (USA).

Рукопись поступила в издательский отдел
31 августа 1981 года.