前　言

系统要求

COBOL要求至少192K字节内存。要执行大的程序可能还需要一些额外的内存。应限制每个MS-COBOL程序不多于64K代码。一个MS-COBOL程序系统可由一个主程序和一个或多个子程序组成，总的代码可以超过64K，但每个程序必须限制在64K以内。整个系统的工作数据存贮区限制在大约60K左右。

有关手册

下列手册提供COBOL的说明：

“COBOL编译程序用户指南”

提供了一个特殊实现或操作系统的一些情况，包括一系列系统要求和磁盘内容说明。用户指南也提供了有关在你的操作系统下编译、装入、执行程序的一般指令。

《Microsoft COBOL参考手册》

该手册对Microsoft COBOL进行了详细说明。该手册介绍的情况适用于大多数Microsoft COBOL编译程序具体实现，例外情况在用户指南中专门有注释。

《Microsoft COBOL快速参考指南》

大致勾画了COBOL程序的结构，并给出了单条语句的格式。

目　　录

第一章 简 介

Microsoft COBOL编译程序（MS-COBOL）版本2.0，运行于XENIX环境，能接受按Microsoft COBOL参考手册定义的语言所写的程序。Microsoft COBOL语言遵循ANSI X.23-1974COBOL要求。

上千种为大型计算机写的COBOL应用程序可以改编和转换成Microsoft COBOL程序。

Microsoft COBOL编译程序不仅为用户提供了COBOL的高级程序设计语言的优点，而且对标准COBOL语言作了许多扩充。该用户指南的目的就在于帮助用户建立COBOL程序并在计算机上运行。

1.1 特性和利益

1.Microsoft COBOL能够定义完整的交互式终端屏幕，并且允许单个语句的数据输入和显示。MS-COBOL支持如下特性的交互数据输入：

a.自动光标定位；

b.自动的数值域编辑；

c.诸如反相的视频方式(REVERSE-VIDEO)和强亮度(HIGHLIGHT)等数据域屏幕特性。这些能力来源于对数据部（屏幕节）和过程部（ACCEPT和DISPLAY语句）的扩充。（参看Micosoft COBOL参考手册6.6.4节“屏幕节”以及第七章“过程部”的ACCEPT语句格式1、3和4及DISPLAY语句）。

2.格式3有COMP-0，COMP-3和COMP-4三种数据格式。COMP-0和COMP-4分别是两个字节和四字节二进制整数。COMP-3格式把数压缩成两位放在一字节中。这三种数据类型都可用来减少数据部内存要求、减少数据文件存贮量要求和加快某些操作的执行速度。

3.小写字母都当作大写字母来处理，非数值直接量在引号中的那一部分除外。

4.用动态调试语句READY TRACE、RESET TRACE和EXHIBIT可在程序执行时显示过程名或数据项。交互式调试工具（一个标准的COBOL特性）提供了功能很强的运行时调试功能。

5.环境部的SELECT子句支持有RECORD KEY和ALTERNATE RECORD KEY子句的分离关键字选择项。

6.CHAIN语句和CHAINING短语扩展了程序间通信的范围，允许把任何程序调入内存并执行。

7.文件共享结构支持在多用户/多任务系统中的文件处理。在OPEN、READ、START和UNLOCK语句和SELECT子句中使用了新语法格式。

8.对分类文件通过在文件控制款中SORT STATUS子句实现分类文件状态报告。

9.第八章“程序间通信”中描述的EXIST、RENAME、REMOVE、COMMAND、UPCASE和LOCASE嵌入扩展子程序提供了一般COBOL程序没有的附加工具。

Microsoft COBOL支持四种类型的文件：顺序的、行式顺序的、相对的和索引的。

Microsoft COBOL索引文件是由Microsoft索引顺序随机存取工具建立的，可用重建工具恢复。包含在Microsoft COBOL里面的Microsoft索引顺序随机存取工具提供了对有索引的数据记录的快速随机检索。

1.2 怎样使用该指南

该指南提供了当用MS-COBOL编译程序编译和运行MS-COBOL程序，以及用运行时执行系统（runcob）执行被生成的代码时的信息。

第一章大致说明了Microsoft编译系统的能力。

第二章到第四章说明怎样编译、装入和执行MS-COBOL程序。这些章节中也给出了关于你的MS-COBOL具体实现的特殊情况 。

第五章解释了COPY文件的搜寻路经的定义。还包括在COPY语句中使用库名和在编译命令行用带全路径名的－S开关的讨论。

第六章对四种文件组织方式进行解释：顺序方式、行式顺序方式、索引方式和相对方式。还说明了怎样使用磁盘输入/输出文件和其它类型的文件。

第七章告诉你怎样使用交互调试工具更正程序运行错误。

第八章对用CALL和CHAIN进行程序间通信进行解释。此外还包括了对MS-COBOL扩展子程序的解释。

第九章介绍重建工具。用Rebuild（重建工具）2.0可恢复索引数据文件的对应关键字文件。

附录A强调Microsoft COBOL2.0的新特点。

附录B描述了MS-COBOL编译程序的五个阶段。如果你的程序产生“编译阶段错”该附录就会非常有用。

附录C给出了Microsoft COBOL编译系统（cobol）、执行系统（runcob）、交互调试工具（debugcob）和重建工具（rebuild）的一些信息。

附录D描述了设置MS-COBOL执行系统所需的终端特性。

附录E说明与MS-COBOL编译程序一起提供的演示程序。一个终端接口的测试程序、一个简单的MS-COBOL程序和三个演示MS-COBOL屏幕和索引文件处理能力的程序。

附录F列出了MS-COBOL错误信息。编译错误分四节按字母顺序列出：命令输入和操作系统输入/输出错误、程序语法错误、文件使用错误和警告信息。运行错误单独以字母顺序列出。

附录G说明了基于特殊微处理器的编码要求，包括两个C语言子程序例子说明基于微处理器数据存贮要求的参数传递方法。

1.3 符号约定

下列正文符号约定在整个手册中用来代表程序设计语法元素。对语句的语法要求可在Microsoft参考手册第七章“过程部”中找到。

[ ] 方括号代表所括内容是可选的

{ } 花括号表示用户可在两个或多个项中选择一个。除非花括号外又套有方括号，否则至少得选一项。

花括号也能定界语句中由省略号涉及的部分。

| 竖线分隔花括号中的选择项。

… 省略号代表一个项可重复任意次。

斜体字 斜体字表示用户必须输入的内容。例如CALL *filename*提示用户输入一文件的名字。 input/output 代表输入和输出及计算机显示的正文。字符应该完全按给出的样子输入。

下划线字符 下划线字符用来增强文件名的可读性。

斜体大写字母 斜体大写字母代表关键字或组合关键词。

所有其它标点符号，如逗号、分号、斜杠和等号等，必须按原样输入。

1.4 对COBOL有进一步的了解

如果你对COBOL程序设计比较陌生，想学习关于COBOL语言的更多东西，那么，下列COBOL教科书是为初学者准备的：

Abel, Peter COBOL Programming:A Structured Aproach, Reston, VA: Reston Publishing CO, 1980.

McCracken,Daniel D. A simplified Guide to structured COBOL Programming.New York:John Wiley and Sons, Inc., 1976.

Parkin, Andrew. COBOL for Students.London:
Edward Arnold Ltd., 1975.

Seidel, Ken. Microsoft COBOL.Beaverton,OR.: Dilithium Press, 1983

Welburn,Tyler. Structured COBOL-Fundamentals and Style.Palo Alto, Mayfield Publishing CO. 1981.

第二章　系统初启

该用户手册的目的在于帮助你建立和执行你自己的Microsoft COBOL程序。为此，不仅应学习了解使用MS-COBOL的主要步骤，还应该做两项初始化工作。这一章里首先列出了COBOL系统软盘上的内容，然后说明怎样进行系统安装和设定终端配置。系统安装和终端配置这两项工作一般只需做一次。最后一节是开发一个应用程序，编译和执行的简要过程。

2.1　Microsoft COBOL系统软件

该编译系统共有几张盘，第1张盘上有一个名叫files.doc的文件，它包含了哪张盘具体装了什么内容的信息。

注意：

第1张盘上文件files.doc，README和update.doc可能装了些印该手册时还没有的信息，请在使用MS-COBOL前找到它们，并认真地读读它们的内容。

你的COBOL编译系统盘上装有下列文件：

MS-COBOL编译程序

cobol	COBOL编译程序
cobol0.ovr	覆盖块0
cobol1.ovr	覆盖块1
cobol2.ovr	覆盖块2
cobol3.ovr	覆盖块3
cobol4.ovr	覆盖块4

执行系统

runcob	COBOL执行系统
debugcob	COBOL交互调试工具

非COBOL子程序库

mknowcob	用来连接非COBOL子程序的shell文件
coblib1.a	装有运行时各例行程序的库
coblib2.a，coblib2debug.a	
coblib2.a	包含所有其它运行时例行程序的库
usrprog.c,usrprog.h	把非COBOL的例行子程序纳入MS-COBOL运行时执行系统所需的模块

实用程序

terminstall	终端屏幕设置程序
rebuild	索引文件恢复程序

演示程序

center.cob　　COBOL演示程序
crtest.cob　　COBOL CRT驱动器测试程序
center.int　　Center.cob的编译好的目标代码文件

MS-COBOL演示系统

demo.cob　　一个COBOL程序，演示MS-COBOL屏幕节调用子程序build，然后再链接到程序update
demo.cpy　　在demo.cob中，COPY动词所使用的文件
build.cob　　一个用来建立名字、地址和电话号码的索引文件的子程序
update.cob　　显示和修改由build建立的索引顺序存取文件

其它文件

README　　含有安装信息的文件
files.doc　　记录各盘的内容
update.doc　　记录修改信息的文件

2.2 在你的系统上安装Microsoft COBOL

第一张编译系统盘上有一进行自动安装的Shell程序。这节将对怎样在你的硬盘上安装该编译程序进行解释。

要安装Micrsoft COBOL编译系统你必须具有超级用户的权限。请按如下步骤安装：

1.注册到根目录上。

2.用cd 命令把当前目录改到/tmp：

```
cd /tmp
```

3.把第一张COBOL盘插入软盘驱动器并关好驱动器门。用tar命命从磁盘上提取安装程序msinstall：

```
tar xvf /dev/fd048ds9 msinstall
```

4.在系统提示符下打入：

```
msinstall
```

并按回车键执行安装程序。

msinstall 程序自动从磁盘上提取文件并装入适当的目录下。当你看到如下信息：

```
Next disk [y, n] ?
```

插入下一张磁盘，打“y”并回车。对软件包的每张盘都这样做一次。当所有盘都安装完毕，对“Next disk”提示回打“n”退出。

5.打入：

```
rm msinstall
```

删除安装的信息。

Microsoft COBOL现在就可以用了。从软盘驱动器中取出最后一张盘，把这些系统盘保存在安全之处，以后如果硬盘系统被破坏的话还会需要它们。

2.3 终端配置设定

MS-COBOL运行系统通过从终端性能数据库（文件'/etc/termcap'）中抽取适当信息在运行时对自己进行设定。

TERM环境变量的值用来从termcap文件中找到正确的终端登记项。用如下命令设定TERM环境变量的值

```
echo $TERM
```

将会在屏幕上显示你的终端特性。

如果你要使用vi或ex编辑程序，你的termcap文件已经被设置成能处理大多数CO-BOL屏幕功能。若要使用所有可能的屏幕特性，包括HIGHLIGHT（强亮度）、REV-ERSE-VIDEO（反相的视频方式）和彩色，或者你的终端不包括在termcap终端特性设置库中，就必须手工或用terminstall程序修改termcap文件。请参考附录D“安装终端接口”中关于编辑信息、功能键说明和关于terminstall程序的介绍。

注意：

termcap文件设置成能处理大多数终端，若当你执行MS-COBOL程序时，终端屏幕上的信息杂乱无章。就需要在termcap中加入或修改有关你的终端的登记项。

2.4 编译和执行

MS-COBOL程序必须先编译后执行，MS-COBOL编译程序由主模块（cobol）和5个覆盖模块（cobol0.ovr到cobol4.ovr）组成。编译程序先对你的COBOL源程序进行分析，然后产生目标代码文件，目标文件的扩展名为.int。

编译按两遍进行。第一遍产生程序的一种中间代码形式，存放在一个临时工作文件中，第二遍生成目标代码的最后形式。运行时系统(runcob)调入并执行编译好的程序。

图2.1简单地勾画了“编译、装入和运行”的过程。

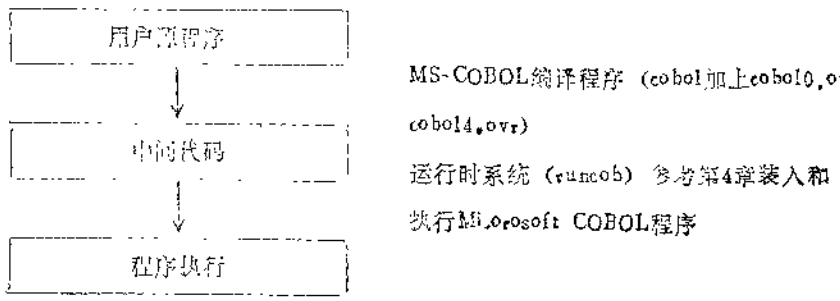

图 2.1 编译和执行MS-COBOL程序的主要步骤

2.5 程序开发过程实例

下面提供了一个在XENIX系统下开发MS-COBOL程序的简单过程。这里用系统盘中提供给你的enter.cob演示程序为例，你也可建立自己的源程序。

先用快速编译检查语法错误。对源程序进行编译并把错误和源列表显示于屏幕上：

```
cobol -l -o  center
```

因为未产生目标和源程序清单文件，编译速度比平常快。

如果试编译期间出现错误，请参考附录F中MS-COBOL出错信息一览表。

要产生目标文件（名字为center.int）的话，请用如下命令：

cobol center	产生目标文件和调试文件
cobol -l center	产生目标文件和调试文件、并在屏幕上列出清单
cobol -L center	产生目标文件、调试文件和清单（center.lst）

注意：

可用-d选择消去调试文件，但我们建议你产生并保存这个文件以便使用MS-COBOL交互式调试工具。

当编译成功地执行完，显示如下信息：

NO Errors or warnings

编译程序结束执行，返回操作系统。

即使你非常小心地排除了所有的编译错误，程序执行期间出现运行错误仍然是可能的。出错信息在本手册附录F中有说明。如果出现运行错误，需要重新调用编辑程序来更正错误。

第三章　编译Microsoft COBOL程序

3.1 MS-COBOL编译程序要求的文件

要编译MS-COBOL源程序需要如下文件

/usr/bin/cobol	MS-COBOL编译程序
/usr/lib/cobol0.ovr	MS-COBOL编译程序覆盖模块
/usr/lib/cobol1.ovr	MS-COBOL编译程序覆盖模块
/usr/lib/cobol2.ovr	MS-COBOL编译程序覆盖模块
/usr/lib/cobol3.ovr	MS-COBOL编译程序覆盖模块
/usr/lib/cobol4.ovr	MS-COBOL编译程序覆盖模块

这些文件可放在你的shell查找路径的任何目录中。shell将用该路径找到cobol，cobol也用此路径寻找其覆盖模块。如果在查找路径中未找到覆盖模块就在“/usr/lib/cobol”中查找。

3.2 调用编译程序

MS-COBOL编译程序读入你的源程序并产生一个列表文件和一个目标文件。命令行的语法包括（1）编译程序的调用和（2）你的命令串。

一旦调用了编译程序，则编译程序被从磁盘上调入，接着检查命令串。并发现有错会在屏幕上显示适当的错误信息。

你的命令串应该象这样：

```
cobol [-switch (es) ] sourcefile [.cob]
```

“cobol”调用编译程序，'[Switch (es)]'代表编译开关可选项，'sourefile [.cob]'是源程序文件名。

文件名都可以是磁盘文件名或系统设备名。文件名的完整说明与你的操作系统有关。源程序文件名存放在磁盘上要用.cob扩展名，但在命令行中指定源文件时可省略扩展名。

编译程序产生的文件的名字用如下扩展名代替源代码文件的.cob：

.int	目标文件
.dbg	调试文件
.lst	列表文件
.nnn	目标文件覆盖块；nnn是三位十进制数字

MS-COBOL编译程序首先在当前目录寻找其覆盖块（cobol0.ovr到cobol4.ovr）如果覆盖块未找到，就找你的XENIX PATH环境定义的路径，接着再找/usr/lib/cobol目录。如果覆盖块仍然未找到就报告出错。用户自己可用-c开关指定可以找到覆盖块的目录。

当编译结束，程序会告诉你有关的错误信息。如果有错，你必须在源程序中找出错误并更正之，然后重新编译。如果编译程序未发现任何错误，就会提示

No Error or warnings

你可以接着做下一步工作了。

3.2.1 文件名

编译时源文件、目标文件和列表文件的名字可以是磁盘文件。MS-COBOL希望是如下格式

path filename extension

path要求是有效的XENIX 路径名。请参考XENIX操作系统用户指南中关于路径的说明。如果未指定路径，则使用当前目录。

filename 是在磁盘上的文件的名字

extension 作为文件名的后缀，用圆点'.'跟一到三个字符表示。若未指定扩展名，暗指：

.cob 源程序文件

.int 目标文件

.lst 列表文件

3.2.2 编译开关的使用

可在编译命令串或任何交互的回答后加上一个或多个开关。开关用连字符(-)标出。MS-COBOL编译程序支持以下开关：

-c *directory* 寻找编译程序覆盖块的目录。

一般，编译程序在当前目录寻找五个覆盖模块，如有必要再找XENIX PATH变量指定的目录和/usr/lib/cobol。用-c编译开关可指定寻找覆盖模块的目录。例如：下列命令行指定编译程序到/usr/leon/work目录下去找覆盖cobol -c /usr/leon/work text.cob

-d 停止产生调试文件。

该开关要求编译程序既不要产生调试信息文件（.dbg扩展名）也不产生放在目标文件中的源行号。结果为压缩的过程部代码（大约紧缩16%）。当然，用了此开关后，运行时系统就不能指出错误产生的行号（请参看第七章"交互式调试工具"）。

例如：

```
cobol -d center
```

目标文件中就不包括源程序行号，也不生成center.dbg文件。

-fn要求FIPS标记

该开关可用来请求编译程序对每条超过美国信息处理标准（FIPS）（n）级的语句给出一个警告。n必须是数字0到4（隐含为4）。

0 对低级以上的任何内容都给标记

1 对中低级以上的任何内容都给-标记

2 对中高级以上的内容给标记

3 对高级以上的内容给标志

4 无标记

-g 只编译带"D"的语句

该开关指示只编译第七列为“D”的程序语句，其余的都处理成说明。它与在ENVIROMENT DIVISION（环境部）中SOURCE-COMPUTER（源计算机）段指定WITH DEBUGGING MODE有相同的效果。

-l 在屏幕显示终端（标准输出）上产生清单。

-L 在文件（源文件名.lst）上产生源程序清单。

-o *file* 产生一用指定名字的目标文件。

-O 不要产生目标文件。

-p n 把页长设成n行。n必须是10到100的整数。隐含的页长是66行。

-s *directory* 把搜索COPY文件的缺省目录改成指定目录。

-t 恢复MS-COBOL前一版本中使用的8，12，20，28，36，44，52，60，68和73制表设置。

3.3 源程序清单文件

源程序清单文件是对源程序进行的一行行解释，而且包含页头信息和错误信息。每一源程序行冠以一个行号，供指出哪行有错参考。

通过源程序中COPY语句指定的文件也包括在清单文件中。

编译错误信息在清单文件尾部给出，并在终端上显示。请参看附录F关于错误信息的解释。

3.4 大型程序的编译

偶尔，一个MS-BCOBOL程序太大以致不能在可用的空间内编译。为避免在编译时发生这种情况，可以把程序分成几个程序模块。这些模块可分开编译，然后作为一个程序系统运行。可参考第八章“程序间的通讯”中关于程序模块的使用说明。也可以把大程序分成几个独立的能分别编译的链在一起的小程序。

另一办法是从磁盘上删去编译程序的中间文件STX$$$、DTA.$$$和LNK.$$$以获得更多的磁盘空间：用lc操作系统命令（在XENIX操作系统用户指南中有介绍）可检查磁盘内容，用如下命令可删除这些中间文件：

```
rm *.$$$
```

第四章　装入和执行MS-COBOL程序

一旦成功地编译完MS-COBOL程序，可用执行系统装入并执行你的程序。

打入“runcob”后跟目标文件名（扩展名.int可省略），执行系统就会调入你的程序并执行。例如：执行演示程序center.int可打入：

```
runcob center
```

4.1　寻找.int文件

运行时的执行系统一般用XENIX PATH环境寻找目标文件。所以如果你的路径（PATH）是

“.:/bin:/usr/bin:/usr/leon/bin:usr/cobol/bin”

运行时的执行系统会检查你的路径中指定的目录来寻找.int目标文件。该规则有两种例外情况：

1.如果目标文件名以显式的路径符号(1)开始，则只查找该路径指定的目录。例如，

```
runcob /test/test1
```

在目录/test中寻找执行文件test1.int。

2.如果有-s运行开关跟一有效路径名，该路径名成了缺省查找路径。例如

```
runcob -s /usr/bobz/work/bin test
```

在搜索路径/usr/bobz/work/bin上寻找test.int。

这些搜索规则也适用于被MS-COBOL CALL语句调用的文件的装入和执行，以及由CHAIN语句装入的文件。

4.2　运行开关的使用

可以把运行开关放在命令行中目标文件名的前面。带运行开关的运行命令格式是：

```
runcob   [-开关]          文件名      [.int]
```

开关

-p　把输出文件送给系统假脱机打印系统。

如果未用-p开关，打印文件被加到当前目录下的print_spool文件后面。例：

```
runcob -p test1
```

把打印文件直接送往系统打印机。

-s *pathname*　替换目标文件的搜索路径。

该开关为查寻目标文件提供代换的搜索路径。如果在当前目录中未发现目标程序，执行系统将在指定路径下寻找。参考4.1节“寻找.int文件”。

-v　修改格式2的ACCEPT语句。

该开关改变格式2的ACCEPT语句（在Microsoft COBOL参考手册7.6.1.2节中有描述），使之象处理字母数字型的一样处理数值型的接收字段，以便与ANS1-74 COBOL标准相适应。

第五章　COPY文件和程序库

本章说明了怎样用库文件名限制COPY语句文件名，以及怎样用库名和路径查找COPY文件。

COPY语句是一个指引语句而非可执行语句。它逻辑地把磁盘文件的内容（非源文件）嵌入源程序中，可用于环境、数据和过程部的任何地方。COPY语句的一般格式是：

COPY 文件名 [{ OF | IN }库名]

注意

下面的讨论也适用于带有REPLACING短语的COPY语句

5.1 路径

路径是引导编译程序搜索文件的一系列用斜杠(/)分隔的目录名。如果路径的第一个字符是斜杠，那么搜索从根目录开始；否则从缺省目录（一般为当前工作目录）开始。例如

/test/cobol　从根目录开始

new/cobol　从省缺目录开始（如果缺省目录是/usr，new/cobol的全路经则为/usr/new/cobol）。

对COPY文件来说缺省目录一般是当前目录MS-COBOL编译命令行的-s开关改变COPY文件的缺省目录。

请参考《XENIX操作系统用户指南》关于树形目录结构和路径的说明。

5.2 给COPY文件指定路径

如果文件名中未带路径，MS-COBOL编译程序从省缺目录中查找COPY文件。省缺目录一般是指当前目录，所以当编译程序遇到一个如下的COPY语句：

```
COPY file1
```

时，它从当前目录中搜寻file1。如果COPY文件名与一条路径一块给出：

```
COPY /test/file1
```

编译程序将在目录/test中搜寻文件file1。

如果MS-COBOL编译命令行中用了-s开关，供COPY文件搜寻的缺省目录可能被改变。

例如　假如你的COPY文件file1在目录/test中，/test也是/new/cobol的一子目录。MS-COBOL命令行通知编译程序关于文件位置的初始信息是：

```
COBOL -s /new/cobol sample
```

sample.cob中包含这个给出文件最后位置的COPY语句：

```
COPY test/file1
```

编译程序通过使用下列成分构造路径来搜寻file1：

—命令行指定路径 /new/cobol

—COPY语句指令路径test

—文件名 file1

最后搜寻到file1的路径是：

/new/cobol/test/file1

例一把当前目录当省缺目录时的COPY文件。

对下列例子，假设：

1.当前目录是/test

2.cobol命令行是

cobol sample2

（命令行中未指定省缺目录）

下表中，每一COPY语句给了编译程序寻找file1的一些或所有路径信息。编译程序将用指定的文件名（左列中）和当前目录值产生全文件名（右列中）。用任一这样的文件名都可使编译程序找到file1。

COPY语句	结果文件名
COPY file1	/test/file1
COPY cobol/file1	/test/cobol/file1
COPY /cobol2/file1	/cobol2/file1

在这个例子中，当前目录从当前系统值/test中获得。

例一指定缺省目录的COPY文件

假设Cobol命令行是

cobol -s /new sample3

指定缺省目录为/new。文件file1的结果文件名将是：

COPY语句	结果文件名
COPY file1	/new/file1
COPY cobol/file1	/new/cobol/file1
COPY /cobol2/file1	/cobol2/file1

缺省路径只有在COPY文件名未指定从根开始的路经时使 用。当指定了从根开始的目录时，则指定文件名中的值将取代目录。

5.3 把程序库名作为路径给COPY文件命令

COBOL允许COPY语句使用的文件名被限定于一个库文件。XENIX操作系统下的MS-COBOL下，程序库名作为XENIX子目录结构的一部分来实现。

当下列格式的COPY语句遇上程序库名字被指定的情况：

COPY 文件名 [{ OF | IN }库名]

编译程序把程序库名作为文件名的直接目录。

使用库名的COPY文件的全路径是先用库名再加一斜杠后接文件名。

例：一带库名的COPY文件

对于下面的例子，假设