大型计算机

应用技术培训

CICS 联机程序开发与调试

目录

课程介绍

CICS 联机程序开发与调试

目的：

本教材的目的是通过对教师的讲解和学员的具体操作，通过习题的练习，使学员能够达到能够对 CICS 系统建立明确的概念，依靠技术文档的帮助，独立地进行 CICS 应用程序的设计、开发、测试和维护工作， 能够比较全面地掌握程序开发员和程序测试员在进行 CICS 程序中所应当掌握的各方面的技能，并能够对立地借助相关的技术文档不断提高自己的能力。

主要内容：

学员主要完成以下主要内容的学习：

- 学习了解 CICS 体系结构
- 学习 CICS 嵌入式程序的编译与排错
- 学习 CICS 基本 API 编程
- 学习 CICS 读取和更新外部数据
- 学习任务和存储管理
- 学习 CICS 队列的使用
- 学习 BMS（Basic Mapping Support）程序设计
- 学习使用 CEDF 进行程序调试
- 学习 CICS 常用的资源定义与操纵交易使用
- 学习 CICS 提供的其它常用交易

预修课程：

IBM 大型计算机基本操作

COBOL 语言程序设计

DB2 与 SQL 编程

长度：

3 天

相关课程：

CICS 系统管理

CICS 性能调整

DB2 与 SQL 编程

教程作者：温洪涛 venn@sina.com

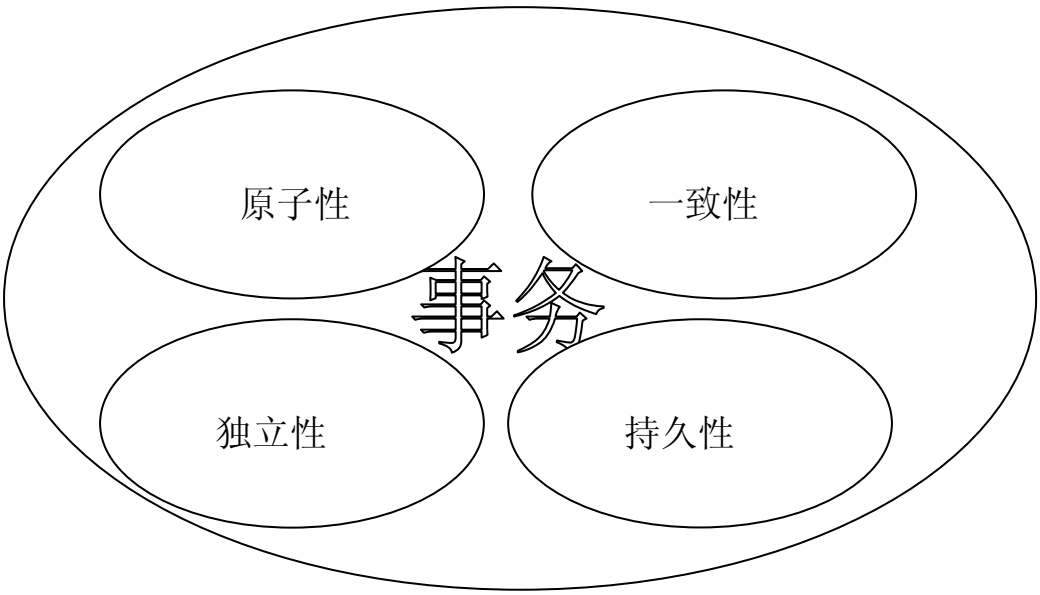
第一章 CICS 概念与功能

- CICS 相关的几个重要概念
- CICS 的主要功能
- CICS 数据管理
- CICS 通信管理
- CICS 应用程序开发

1-1 CICS 相关的几个重要概念

CICS（Customer Information Control System）是 IBM 公司开发的联机事务管理系统，这里我们首先介绍相关的几个重要概念。

1-1-1 事务（Transaction）



事务（Transation）：

事务就是对于某种要求进行处理并返馈结果的一个完整处理过程。在这种类型的处理程序中，有大量的相似的处理要求，联机事务管理系统就是提供事务处理中需要的通用逻辑处理功能，使用户的业务处理程序能够专注于应用本身的逻辑的软件系统。

事务处理要求的特性：

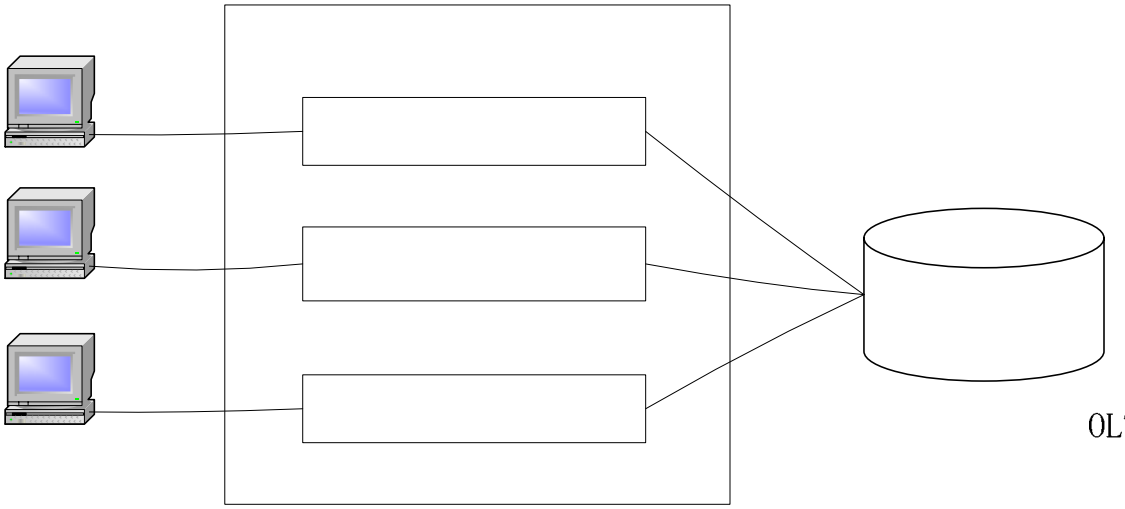
原子性(Automicity)：一个事务中若干相关操作作为一个完整的单元进行处理，一个单元中的所有操作，要么整个完成（Commit），要么就得完全恢复到事务发生之前的状态，好像什么也没发生过一样(Rollback)。

一致性(Consistency)：事务必须在两个一致的状态间移动数据，并以一种可重复的方式操作。如果事务发生前相关数据是一致的，事务完成后数据必须仍然一致。在事务内部因为数据处理的先后，可能出现数据的不一致，但在事务完成后外部数据要达到一致。如果重复某一事物，它总是按相同的逻辑执行。

隔离性（Islation）：各个事务之间可以独立运行，互不干涉。一个事务只能看到另一个事务发生前或发生后的数据，而不能接触另一个事务运行中的数据。

持久性（Durability）：当一个事务完成后，他所涉及的数据能够持久地保持在系统中。这一特性使得在系统发生崩溃时，相关的数据不至于丢失错乱，在系统恢复后能够恢复交易数据。

1-1-2　OLTP

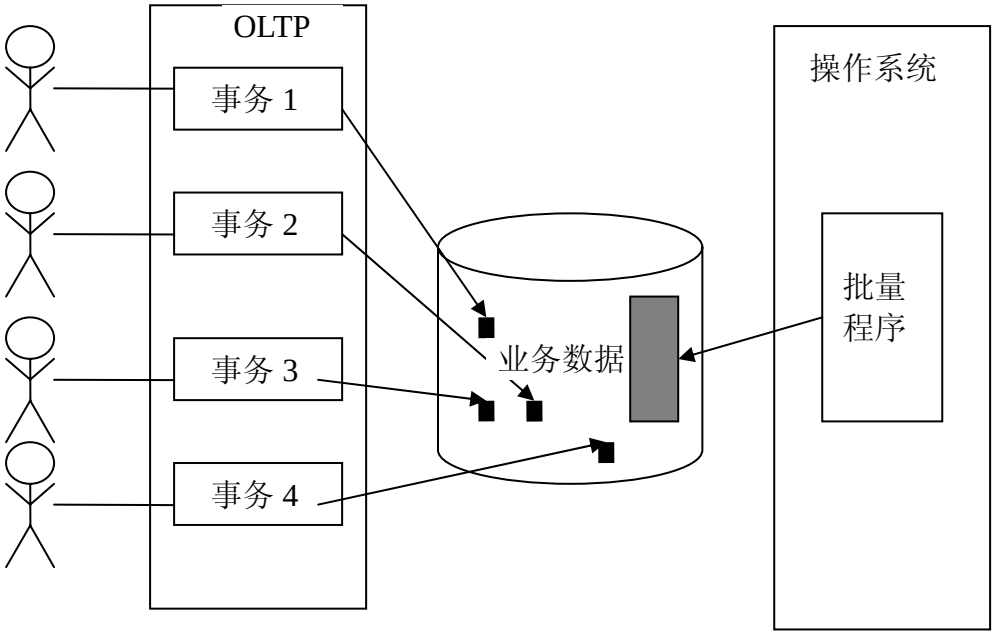


OLTP，（On-Line Transaction Processing）

联机事务处理，是面向事务应用的实际运行模式，其基本形式是大量会随机申请服务的客户，通过多种设备和渠道连接到业务处理系统中来，每一个服务请求在业务处理系统中以一个事务的方式运行，通过对一定业务数据的处理来完成客户的请求，并向客户返回数据。通常，这种应用程序有大量的客户同时执行更改实时数据的事务。尽管客户对数据的单个请求一般只引用少量数据记录，但是，这些请求有许多是同时发生的。OLTP 应用的常见例子是银行业务系统和航空订票系统。

OLTP 系统中，发生的每个业务处理，都是一个事务，它要符合一个事务所必须具有的 ACID 四个特性，只有这样才能保证系统的可用性。

1-1-3 联机程序 批量程序



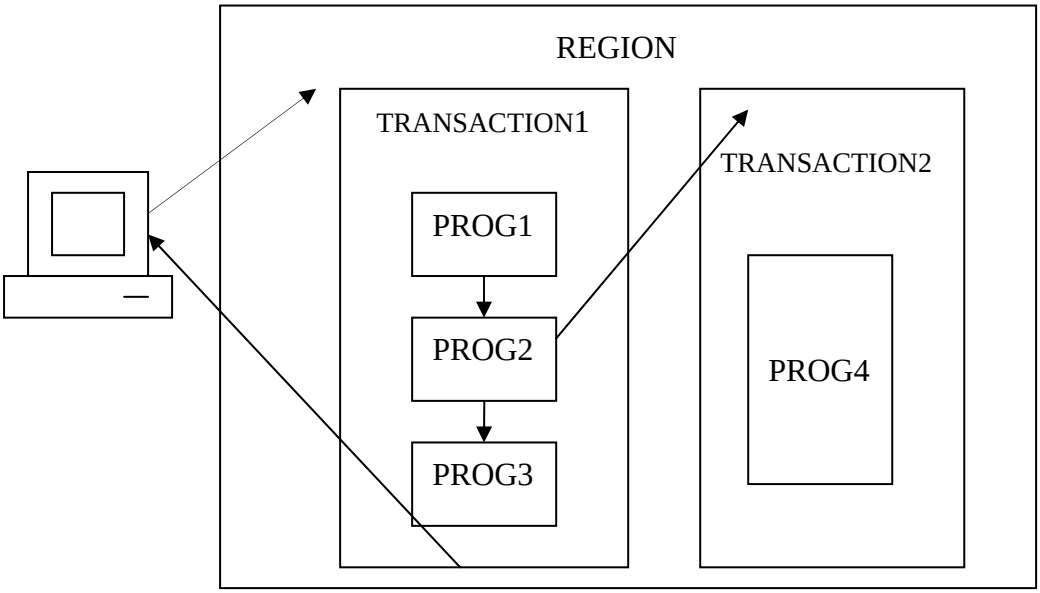
一个综合业务系统，如银行业务系统，程序根据数据的特点不同，一般分为两类运行模式，一类是联机事务处理程序，一类是批量数据处理程序。

联机程序是在某种联机事务管理系统如 CICS 系统的环境中运行的很多小程序，一个或多个小程序组合在一起，以一个事务的形式在运行，它们相应某种特定的请求，完成某种特定的操作。客户的需求是随机出现的，联机应用程序是根据客户的要求随机地被调起，任意一个事件内都有很多个事务在同时地被处理，而每个事务一般只要处理很小一部分数据操作。

批量程序的特点是一次对大量数据进行操作，它一般直接在操作系统的支持下运行，由人工控制启动，运行时间比较长。

传统的业务系统，为了避免不同特性程序互相影响，联机程序和批量程序一般是分时段分别运行的。现代的业务系统，为了提供 7*24 小时的支持，一般要采用各种技术手段来支持联机系统的不间断运行。

1-1-4　区域/事务/任务/程序（Region/Transaction/Task/Program）



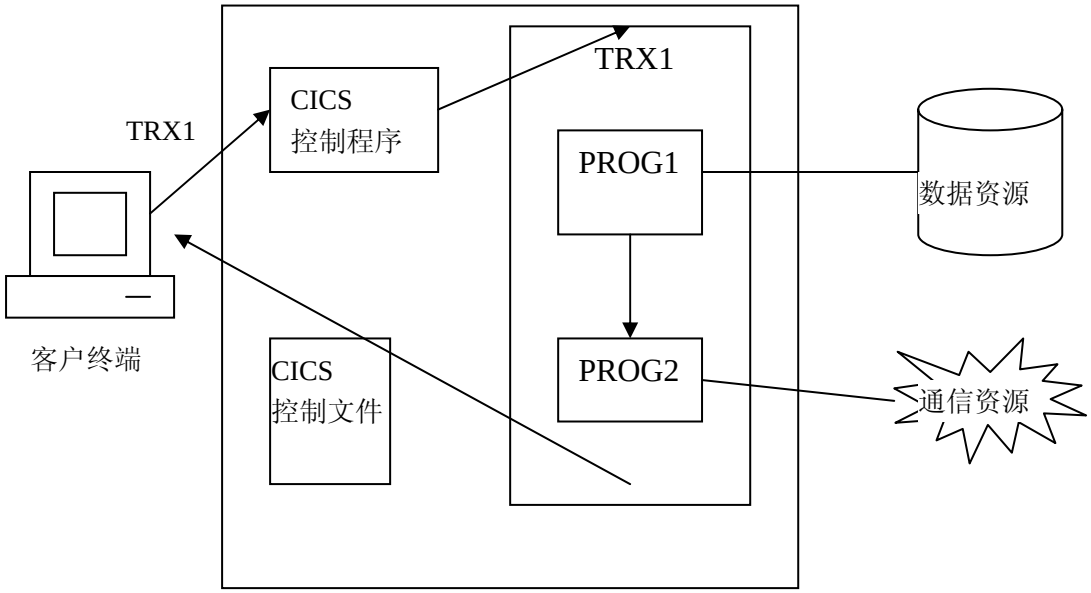
区域（REGION）：CICS 系统的基本单位，一个 REGION 由一组 CICS 系统程序、REGION 的所有配置信息、它所管理的各种资源（交易、程序、数据等等）组成，是一个独立的 CICS 环境。

事务（TRANSATION）：一个事务（交易）用来完成一个特定的业务处理流程，是 CICS 中的一个数据处理单位，一个事务能够完成某项特殊功能。每个事务以一个 1--4 字符的名字标识，分别使用不同的程序完成不同的任务，在一个 CICS 区域中，可以同时运行多个事务。

任务（TASK）：一个任务是事务的一个特定的运行实例。CICS 为每个任务建立一个独立的任务环境，在一个 CICS 区域中，可以同时运行处理同一事务的多个任务。

程序（PROGRAM）：业务处理逻辑的最小单元，一个程序可以完成一段特定的处理，每个程序有一个最多 8 个字节的程序名字。程序通过调用 CICS API 可以控制和使用 CICS 资源来完成自己的任务。

1-1-5　一个 CICS 事务的生命周期



当用户应用程序请求 CICS 中的一个事务时，将该请求 传递到一个将要处理的 CICS 区域在这个区域中，一个事务经历以下的生命周期：

1. CICS 区域接收到一个来自用户应用程序的事务请求。 (区域必须首先验证它能够与该用户的设备通信并且该用户已授权使用此系统。)
2. CICS 区域搜索控制文件中的事务定义表，可获得有关该事务的信息。（在使用一个事务前必须定义一些属性，如事务受请求时将要运行的第一个程序的名称。）
3. 如果事务定义存在，则 CICS 区域把请求分配给一个 TASK 来处理这个事务。
4. CICS 根据任务的定义，为他装载第一个程序，由这个程序根据情况决定是否和如何调用别的程序。
5. CICS 监控任务的进展，为其数据、通信及其他资源的请求提供服务。它也要实行一些必需的后台操作以确保这个任务继续最优地运行，避免与其他任务冲突并保证必需的数据完整性。
6. 当任务完成时，CICS 区域提交（Commit）数据的更改，终止该任务并释放资源以用于其它事务。

同一个事务的几个实例可以作为不同的任务 同时运行。

TO COMMIT, OR TO ROLLBACK: THAT IS THE QUESTION

数据一致性通过对数据的提交与回滚实现。

可恢复资源：

CICS 提供的可恢复资源有：用户文件、数据库、TD Queue 和辅存 TS Queue

逻辑工作单元(Logical Units of Work)：

一个 LUW 就是一个交易里从产生第一个能改变任何可恢复资源的要求开始，到 SYNCPOIN 或 ROLLBACK 命令之间的部分。LUW 是数据一致性的单元，一个 LUW 里的改变要么完全 COMMIT 要么完全 ROLLBACK。

程序控制的数据一致性：

应用程序中把一个完整的单元设计为一个 LUW，所有步骤都正常结束时发 SYNCPOINT 命令，一旦发生错误，使用 ROLLBACK 命令使涉及的资源回复到 LUW 发生前的状态。

CICS 系统控制的数据一致性。

CICS REGION 保证业务系统及其数据总是处于一致状态。万一应用程序或系统发生故障（例如，掉电和计算机系统关闭），REGION 可以自动重新启动自身（如果需要）并恢复故障发生时进程中未完成的工作 (包括数据的更改)。如果它不能提交某一任务的数据更改，REGION 动态地把更改逆序恢复到系统最后在一致状态的那一点。

事务处理恢复服务基于以下信息：

逻辑工作单元 (LUW) 和 SYNCPOINT 纪录

它们提供事务所作的上一次提交的记录。

事务记录（Transaction log）

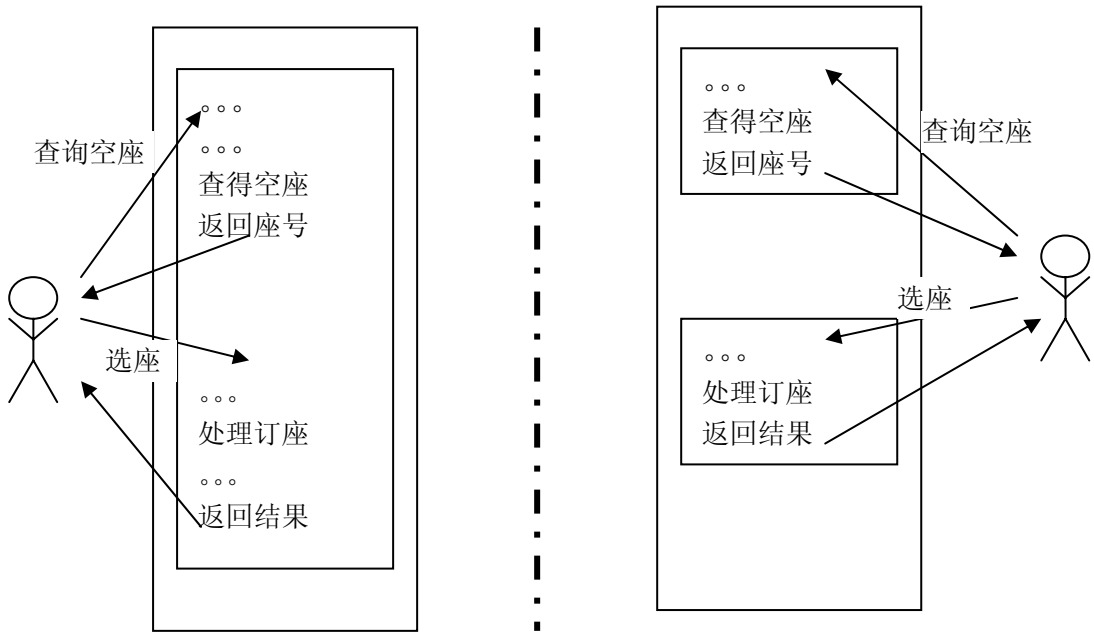
当一个任务正在运行时，CICS REGION 记录它对可恢复数据所作的更改的信息。记录的信息表明了自上个同步点以来未提交的数据更改。

动态事务复原（Dynamic transaction backout）

如果一个任务没能完成，CICS REGION 使用它的记录信息和资源管理器所记录的信息来撤消对未提交数据的更改。这样就将可恢复数据还原到它前一次提交的状态。

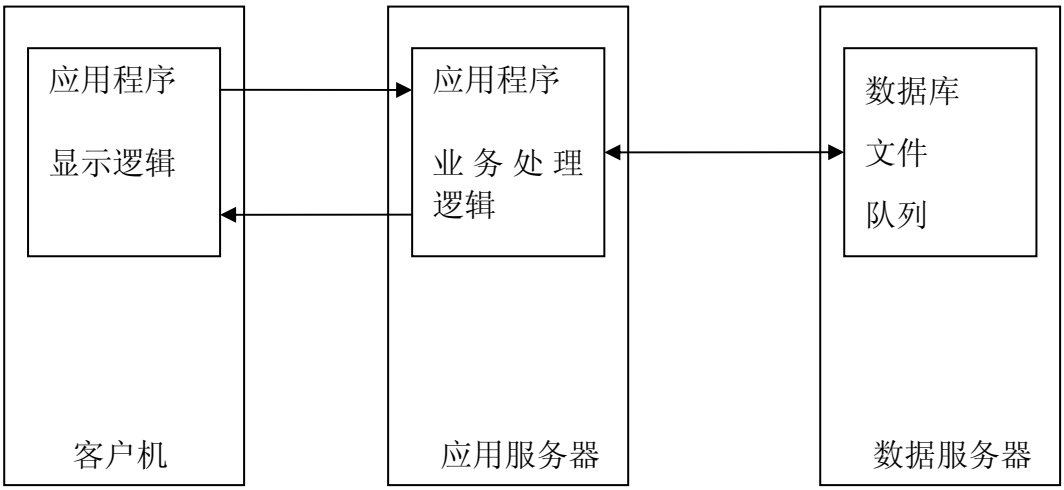
用户日志（User journals）

用户日志记录了不是由 CICS REGION 提供的用于恢复功能的记录信息，例如审查记录。您可以编写应用程序来使用任何用户日志。



会话与伪会话（pseudoconversation and conversation）是应用程序设计的两种模式

RTIMOUT



三层应用结构是把应用程序划分为三个层次，这三个层次的程序通过网络结合起来共同完成业务功能，这三层分别是：

显示逻辑：

显示逻辑用于最终用户和事务处理系统之间的通信。显示逻辑运进行在客户机上，以各种用户友好的方式为最终用户条提供便捷的数据输入和明了的结果显示。

商业逻辑：

商业逻辑层按照事务处理的业务规则加工整理数据，并且执行事务所需的数据操作和计算。

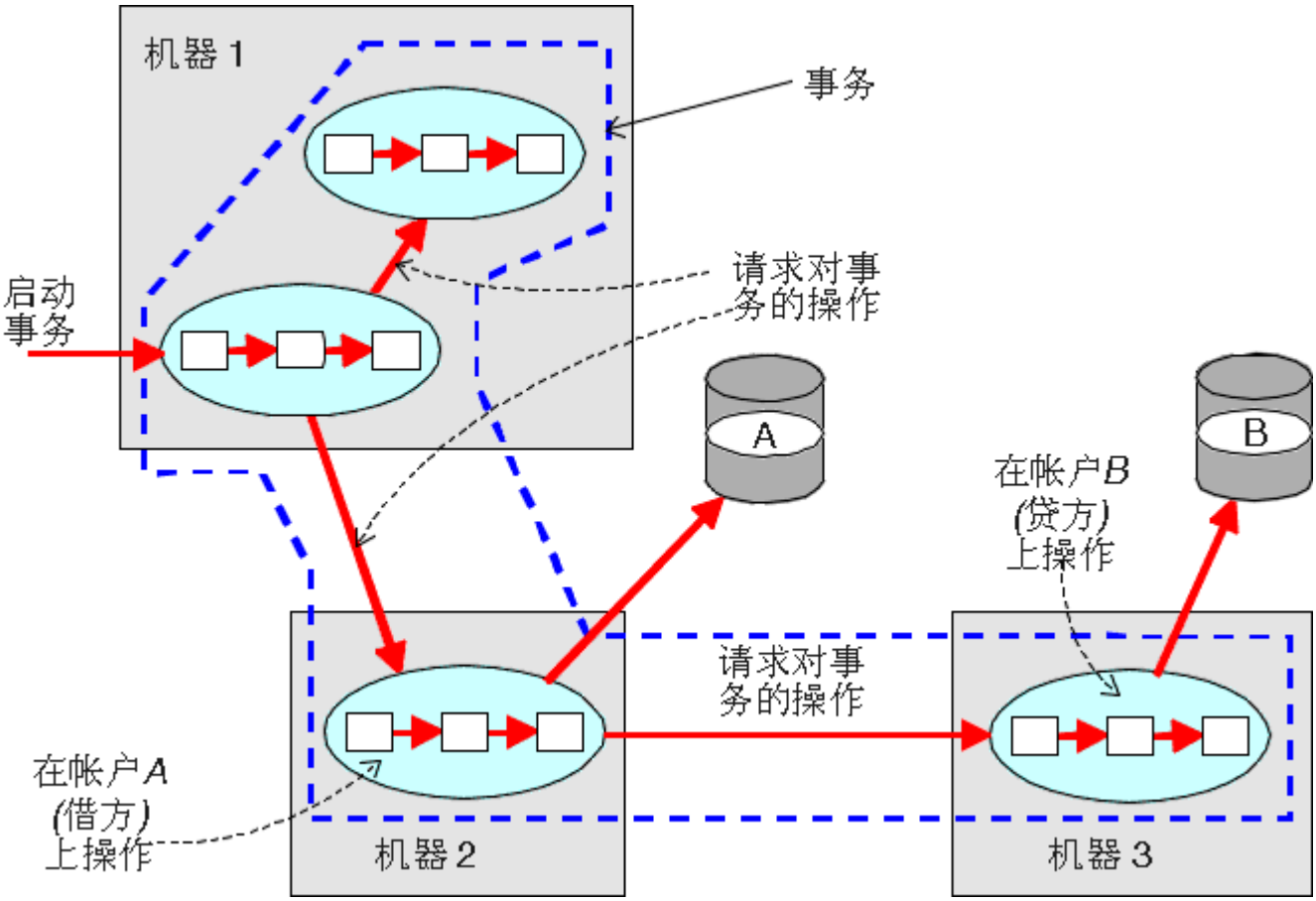
数据服务：

数据服务用于检索和更新数据，由资源管理器提供。

三层结构的优点：

易于维护
可伸缩性
开放性
安全性加强
效率提高

分布式事务处理（Distributed transaction processing）



CICS 提供分布式事务处理功能，一个事务涉及的若干程序和资源，可以分布在不同的 CICS REGION 中，而这些 REGION 可以分布在不同的机器上， 甚至可以是在不同的平台上。

业务应用方面，认为在一个 REGION 上运行和在几个分布式 REGION 上运行的事务之间没有什么功能上的差异。 它仍然对程序、数据和其它资源作相同的调用，并且让事务处理服务器来确定资源的位置。然而，用户可以感受到来自该应用的良好的服务。例如，某个事务处理服务器可优化为用户交互，从而使响应时间加快，并且它能将工作集中的操作传递给另一个事务处理服务器处理。

从系统设计的角度，可将逻辑和数据的各部分放置在它们能被最高效地利用的地方。互连使逻辑和数据能共享。还能通过在不同计算机上提供重复的服务来实现更高的可用性与可靠性。 分布也允许在不影响系统中其他计算机使用的情况下通过增加或更换个别的计算机来实现系统的增长。

1-2 CICS 的主要功能

1-2-1 CICS 联机事务处理的主要功能

- 任务管理
- 程序管理
- 系统资源管理
- 控制数据存取及更新数据
- 数据通信
- 终端管理
- 时间管理
- 安全性管理
- 恢复管理
- 工作负荷分配

任务管理

任务管理服务控制任务的创建、处理和终止。CICS REGION 依照任务的相对优先级、当前 TASK 总数和其它系统资源来调度和分派任务。CICS REGION 控制处理任务的速率和次序，从而减小冲突和系统超负荷的机会。

程序管理

程序管理服务用来将任务与执行其工作的应用程序联系起来。虽然许多任务可能需要使用相同的应用程序，但程序控制只在内存中装入一份代码副本。每个任务各自独立地执行代码，所以许多用户能使用应用程序的同一个物理副本来并行地运行所有的任务。

系统资源管理

CICS REGION 将应用程序从不得不与操作系统协商以获得和释放资源中释放出来。一个正在处理的任务需要与其他任务共享 CPU、申请内存以存放用户数据、需要用于临时计算的内存小区域，并且有时它需要数据通信通道、数据文件和数据库。对于所有的这些，都可通过 CICS REGION 从操作系统得到。CICS REGION 把资源分配给需要的任务，当这些任务完成它们的处理或特别释放一个资源时，它又负责收回这些资源。

控制数据存取及更新数据

CICS REGION 使用户能共享相同的数据。它甚至跨多个机器和平台控制资源管理器的同时存取，并维持数据更新的完整性。例如，它同步更新、记录更改并恢复不确定的更新。还能使用安全性服务来保证只有特许的用户可访问和更新数据。

CICS REGION 控制数据存取和更新，这使得应用程序不需要代码与特定的数据管理器交互，当一个任务的程序请求数据时， CICS REGION 决定适当的数据管理器并为该程序检索数据。当一个任务结束后，REGION 将提供给这个任务的资源释放，以便让其它事务处理任务或其它程序使用。

数据通信

为了使许多用户能同时工作，CICS REGION 必须最大地减少用户等待的时间。它使用 1-2-1 数据通信服务来识别哪个用户发送了一个特定的信息并将数据发送到正确的用户。该 REGION 还保证以与用户设备兼容的方式发送显示数据。

终端管理

终端管理提供设备独立性，使应用程序能以一种标准的方法与任何类型的终端通信。CICS REGION 查询用户的设备并确定用于应用程序输出的最优特性。该 REGION 可以使用模型来影响它的特性选项并用终端定义来应用设备的特殊性能。

时间管理

时间管理服务使程序能启动和控制一系列取决于时间的操作 -- 例如，在一天的某个时间启动一个事务（任务）和在一个指定的时间段过去时发出信号。这些服务还能把以日期和时间为标记的事件记录到磁盘上，用于记帐或确保数据的完整性，并能启用 CICS REGION 的一定程度的自动化。

安全性管理

CICS REGION 提供安全性来阻止非法登录并保护个别资源 (程序、文件等等) 不被其它所有而不是某些用户使用。安全性管理服务提供 CICS 内部安全性、外部安全性管理器、DEC 安全性服务或它们的某些组合执行检查所需的数据。

恢复管理

CICS REGION 保证业务系统及其数据总是处于一致状态。万一应用程序或系统发生故障（例如，掉电和计算机系统关闭），REGION 可以自动重新启动自身（如果需要）并恢复故障发生时进程中未完成的工作 (包括数据的更改)。如果它不能提交某一任务的数据更改，REGION 动态地把更改逆序恢复到系统最后在一致状态的那一点。

工作负荷分配

可以跨越多个 CICS REGION 分配事务以使工作负荷跨越那些 REGION。可以预先定义事务请求是在本地（即在收到请求的 CICS REGION 上）接受服务，传到指定的远程 REGION 在那儿运行，还是动态地传到能运行该事务的任何 REGION。

1-3 CICS 数据管理

- 数据类型
- 文件
- 队列
- 关系数据库
- 日志
- 数据共享和分布

1-3-1 CICS 管理的数据类型

- 文件
- 队列
- 关系数据库
- 日志

文件 -- 永久存储直到被明确删除的数据。

文件是 CICS 中的可恢复资源，CICS 可以管理的文件是本地或远程的 VSAM 文件，可以使用的 VSAM 文件包括三种类型：关键字顺序数据集（KSDS）、输入顺序数据集 (ESDS) 和相对记录数据集 (RRDS)。文件先在操作系统中建立，然后定义到 CICS 的文件资源 FCT 中，应用程序通过 CICS 提供的接口来操纵这些文件，CICS 通过锁和日志的机制来保证数据的一致性。

队列– 一种以先入先出形式存放的数据。

队列分为两种，一种是 TDQ（Transient Data Queue），一种是 TSQ（Temporary Storage Queue）。TDQ 以一个最长 8 个字符的名字标识，每个 TDQ 使用前都要在 CICS 里先定义，TDQ 必须按顺序读，并且每个元素只仅能被读一次（一个元素被读出后，就从队列中移去并且不可再次被读出了），TDQ 的数据是可恢复的。TSQ 也有一个最长 8 个字节的名字，不过它的使用不必预先定义，可在应用程序要写入队列时随时创建，TSQ 中的元素能够顺序读取，也能按指定的序号直接读取，它们可以被读取任意多次而绝不会被移出队列，直到使用命令清除，TSQ 是一种不可恢复的数据结构。队列一般用来存放处理临时数据或把数据从一个任务或一个程序传递到另一个时使用。

关系数据库 -- 存储在 DB2 系统中，由 DB2 统一管理，使用结构化查询语言 (SQL) 命令访问的数据。

日志 -- 一组在发生故障后用来恢复数据更改的特殊用途的文件。

1-3-2 数据共享和分布

- CICS 本地和远程数据
- 数据完整性
- 恢复

CICS 本地和远程数据：

由 CICS REGION 直接访问的文件和队列对该区域定义为本地。如果通过 另一个 CICS 区域访问数据，则该数据定义为远程。CICS REGION 通过 CICS 功能转移(Function Shipping)方式向远程区域发送对远程资源的请求。

通常，将应用程序设计为能访问任意资源，而不必考虑资源的位置，只需按资源的名称引用。每个资源定义都指明了资源是本地或远程的，对于远程资源，定义中指明资源实际所在的 CICS REGION。如果愿意，CICS 应用程序也可在请求资源时明确地命名一个远程 CICS REGION 名。

数据一致性：

数据完整性是与所有共享数据都有关的一个问题。甚至在只有一个用户更新数据时，就有读一致性的风险，因为更新过程中就可能有其它用户读取它。如果不止一个用户能更新数据，就会有更大的数据一致性风险。如果两个任务尝试更新相同的数据项，它们可能互相干扰或用没有反映另一个更改的数据来覆盖数据。写一致性可通过将所有对同一条资源的更新串行化来维护。串行化的实现方法如下，一旦某个任务快件要更新数据时就对该数据加锁，拒绝其他任务对同一数据的更改，直至锁定持有者完成更新并释放该锁为止。读一致性也可以用类似的方式，通过加锁实现。当然加锁意味着延迟，因此完全的读完整性通常是根据情况决定是否必要的。

在 CICS 中，系统保证同一个逻辑工作单元 (LUW)内所有数据操作的完整性。程序设计时要把需要保证完整性的若干操作放在一个 LUW 里，比如，一个的 LUW 里可以去扣两个帐户的款，然后加总到第三个帐户上去， LUW 保证三个操作一起完成或一起回滚。

恢复

在分布式事务处理环境中，任务所更新的资源可能属于多个资源管理器。在这种情况下，当 CICS 区域控制 LUW 期间一个任务失败时，回滚操作会涉及全部的资源管理器

1-4 CICS 通信管理

- 与用户通信
- 多向通信

- 本地 3270 终端
- 3270 仿真终端
- IBM CICS CLIENT

CICS 本地终端：

与主机直接相连的 3270 终端设备。它可用来输入启动 CICS 事务的请求并接收来自 CICS 事务和用户应用程序的 3270 数据流输出。CICS 本地终端是早期业务应用系统中最重要的用户终端设备。

3270 仿真终端：

随着微型计算机的发展，逐渐出现了各种各样的 3270 仿真终端，它不是一个物理的终端，只是通过软件仿真，对外以 3270 数据各式通信，对内以和 3270 终端相似的方式进行显示，3270 仿真终端有和 3270 本地终端相同的功能。

CICS CLIENT：

它是 IBM 开发的一个软件包，CICS CLIENT 支持外部程序启动 CICS 交易并与之进行数据交换，这使得外部程序即能使用 CICS 事务处理所带来的好处，又可以更灵活地处理数据，以更丰富的界面与用户交互，所以这种方式逐渐成为实际业务应用使用 CICS 功能的手段。在 CICS CLIENT 中调用 CICS 交易主要有两种类型的接口 ECI 和 EPI：

外部调用接口 (ECI) ：

ECI 接口使一个在客户机上运行的非 CICS 应用程序能同步或异步地调用 CICS 程序，并与只交换数据，就象调用了一个子程序一样。非 CICS 应用程序使用简单的 ECI 调用一个交易，并将数据块传递给他，只需指定 CICS REGION 和交易名称，无需任何专门的通信代码，CICS CLIENT 就能把交易路由到正确的位置执行，交易结束时把数据块返回给调用程序，此程序就可以进行进一步的处理。

外部显示接口 (EPI) ：

与 ECI 类似，它使一个在客户机上运行的应用程序能调用服务器上的 CICS 事务。区别是 EPI 调用以 3270 终端接口数据各式与 CICS 交易处理程序间传递数据，执行该事务就如同将它从 3270 终端启动，该交易也向客户返回一个 3270 数据流，而客户可以自己的方式将它呈现出来（如使用图形用户界面等）。

- Function shipping
- Transaction routing
- Asynchronous processing
- Distributed program link (DPL)
- Distributed transaction processing (DTP)

功能运送（Function shipping）：

我们可以用功能运送来使运行在一个 CICS 系统中的程序像访问本地程序完全一样地访问远程 CICS 系统中的资源。只要在 CICS 的资源定义中说明此资源是远程资源，给出它所在 REGION 的信息，当应用程序访问此资源时，系统自动把访问请求发送到远程 REGION。FUNCTION SHIPPING 功能可以用来指定远程系统的文件、DL/I 数据库、TDQ 和 TSQ 等资源。

事务路由（Transaction routing ）：

可以用事务路由来使我们像执行本地事务完全一样地执行远程 CICS 系统中的事务。只要在 CICS 的资源定义中说明此交易是远程交易，并给出它所在 REGION 的信息。用户可以不必关心交易到底在那个 REGION 上，都能像在本地一样执行。

异步处理（Asynchronous processing）：

启动一个事务，然后等待直道它返回，这是同步处理。启动一个事务，不进行等待继续处理，这是异步处理。当应用程序不需要远程 REGION 上交易的返回结果时，可以异步地启动事务在另一个 CICS REGION 上独立地运行。远程启动的事务成功与否并不影响启动它的应用程序。

分布式程序链接 (DPL)

DPL 使在一个 CICS REGION 上运行的程序，可以调用（使用 LINK 命令）令一个 REGION 上的程序进行处理，传给它参数并接受它的返回结果。如果被调用的应用程序需要多次访问远程资源，则使用 DPL 来处理，比直接访问远程资源有更高的效率。然而，如果您的事务需要在两个 CICS 区域之间传送大量的数据，请考虑使用 DTP。

分布式事务处理 (DTP) ：

这是所有内部通信设施中最灵活的一种。它允许两个在不同系统中运行的应用程序相互传递信息。所使用的命令映射为 SNA 中所定义的 LU 6.2 映射会话动词。应用程序可在处理过程的任一时刻发送它们所需的任何数据。但灵活就意味着复杂，因为该应用程序将负责设置与远程系统的通信、发送和接收数据并在结束时最终关闭通信。

DTP 是唯一可用来与非-CICS 应用程序通信的 CICS 内部通信设施。非-CICS 应用程序可使用 APPC 协议与一个 CICS 程序通信。

1-5 CICS 应用程序开发

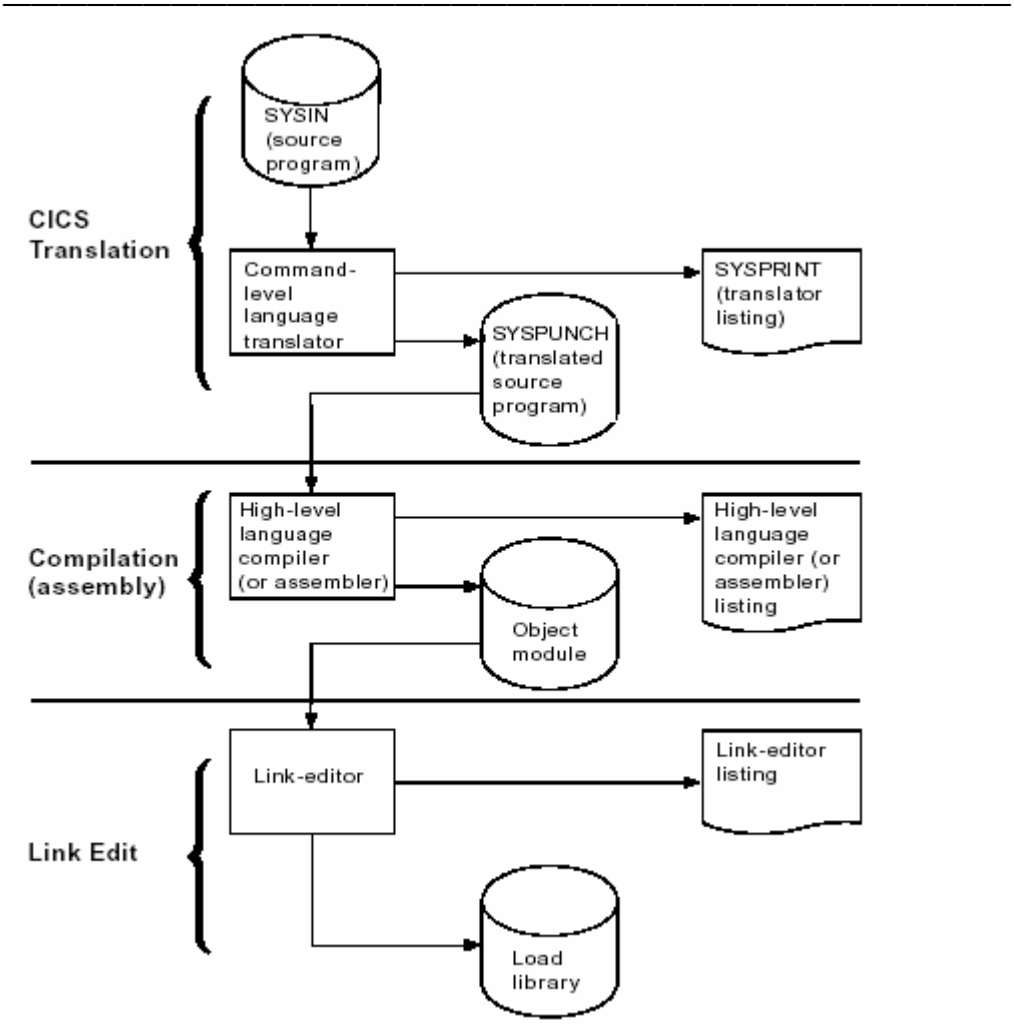
- 开发语言
- 程序开发过程
- 主要 API
- 程序测试
- 外部接口

- COBOL
- C/C++
- JAVA
- ASSEMBLER
- PL/I

就象你能使用多种语言来开发数据库应用程序一样，CICS 应用程序也可以使用多种语言进行开发，从最古老的汇编一直到新式的 JAVA 语言。象 SQL 语句一样，CICS 所提供的功能也是以一定的格式嵌入式地与程序开发语言结合在一起的。CICS 命令都是以类似下面例子的格式嵌入在开发语言中的：

EXEC CICS	SEND	FROM （MY-DATA）	LENGTH （40）	END-EXEC.
关键字	功能	选项 参数	选项 参数	结束标志

1-5-2 程序开发过程



CICS Translation：

通常又称为预编译，CICS 翻译程序把源程序中嵌入的 EXEC CICS 语句翻译为宿主语言的语句和函数。

Compliation：

编译或汇编，这一步骤是把上面生成的高级语言程序或汇编程序编译成为目标代码。

Link Edit：

连接，这一步骤把程序要调用的 CICS 函数库和宿主语言函数连接进目标代码，生成可执行的机器代码。

1-5-3 CICS 应用程序接口

- 终端 I/O
- 访问和修改文件
- 访问和修改数据库
- TDQ 和 TSQ 操作
- 程序和任务操作
- 内存操作
- 通信操作

CICS 提供了一个应用程序向 CICS REGION 请求服务的标准命令集，该命令集称为 CICS 应用程序设计接口（API）。CICS 中运行的程序通过调用 CICS API 的各种命令来使用 CICS 提供的功能。 API 对所有的 CICS 系列产品来说都是公用的， 因此 CICS 应用程序可以从一个平台移到另一个平台。业务应用程序经常使用的 API 命令主要包括终端 I/O、访问和修改文件、访问和修改数据库、TDQ 和 TSQ 操作、程序和任务操作、内存操作等。

1-5-4 CICS 程序调试

- CEDF
- CECI
- CECS
- CEBR
- ...

CICS 为程序开发和调试提供了一系列工具，主要有：

CEDF：它是 CICS 为交互式程序调试提供的一个系统交易，它能够跟踪所有 CICS 命令的执行，察看或修改命令参数，主要用途有：

- ✧ 帮助确定和修正程序中的错误
- ✧ 跟踪每条 CICS 命令的执行
- ✧ 在无源程序是可以用来检查程序所执行的 CICS 指令
- ✧ 允许异常条件的模拟
- ✧ 为异常条件调试提供详细信息
- ✧ 可强制产生内存倾印（Dump）

CECI 是命令解释器。它允许使用或测试一个 CICS 命令而不必编写程序，一般用来配合测试临时修改文件、操作队列等。

CECS 可不实际执行而进行命令语法检查

CEBR 用来浏览、删除 TSQ，常用来检查程序输出的调试信息。

- 外部调用接口（ECI）
- 外部显示接口（EPI）
- 外部 CICS 接口（EXCI）

CICS 的传统交互方式是通过字符界面的 3270 终端，接受客户键入的信息和显示处理结果，随着客户交互技术的发展，越来越多的应用不再满足于简单的字符界面，顺应这种要求，CICS 为外部的应用程序提供调用接口，使提供新型界面的非 CICS 程序能够调用 CICS 事务进行处理。目前主要的接口形式有：

外部调用接口：

外部调用接口（ECI）允许运行于客户机上的非 CICS 应用程序调用运行于 CICS 区域上的事务处理程序。客户程序可以以子程序调用的形式，同步或异步地调用 CICS 交易。客户程序与 CICS 事务处理程序使用一个称为 COMMAREA 的数据区进行通信，该数据区将在调用时被传送到 CICS 区域。CICS 程序通常把处理结果放回 COMMAREA，然后将数据返回至客户机，客户机程序以各种丰富的表现形式显示出来。这种 CICS 程序不能执行终端 I/O，但可以访问和更新所有其它的 CICS 资源。

ECI 能够优化新的应用程序的设计，它的调用结构很容易将显示逻辑（通常位于客户机中）从 CICS 服务器应用程序中的商业逻辑中区分出来，为应用程序设计者提供了最大限度的灵活性。

使用 ECI，客户应用程序以完成：

- ✧ 从一个非 CICS 程序调用 CICS 区域中的 CICS 程序
- ✧ 同时连接至若干个服务器
- ✧ 有若干个未完成程序同时调用
- ✧ 访问 CICS 程序、文件、瞬时数据队列、临时存储器和事务
- ✧ 在客户机和服务器之间交换数据

被调用的服务器可以完成：

- ✧ 在自己的 CICS 区域内更新资源。
- ✧ 使用分布式程序链路（DPL）调用其它 CICS 区域内的程序。
- ✧ 使用功能装运或分布式事务处理（DTP）访问其它 CICS 区域内的资源。

外部显示接口

外部显示接口(EPI)使客户应用程序能够启动运行于 CICS 区域的传统的 3270 CICS 应用程序并与之对话。客户应用程序捕捉这些数据流，然后通常用一种比 3270 更好的显示手段来显示它们。因此 EPI 是一种通过添加图形或其它现代接口以增强现有 CICS 应用程序的方式，这种方式下，即可以提供更好的交互手段，又不必修改原来的 CICS 应用程序。

使用 EPI，可以编写应用程序来完成：

- ✧ 允许 CICS 区域将与其相连的非 CICS 应用程序视为一个 3270 终端来看
- ✧ 同时连接至若干个服务器
- ✧ 有若干个未完成程序同时调用
- ✧ 调度向客户应用程序发送输出的事务

外部 CICS 接口（EXCI）

以上所说的两种接口，是 CICS CLIENT 所提供的接口类型，在 MVS 系统上，非 CICS 程序调用 CICS 程序，要使用 EXCI 接口。MVS 系统上的非 CICS 程序可以通过 EXCI 接口来调用 CICS 程序，并通过一个通信区（COMMAREA）来传递和接受数据。

EXCI 接口提供两种形式的编程接口：

1. EXEC CALL 接口，可以：
   -- 允许 MVS 系统中运行的非 CICS 程序与一个 CICS REGION 建立一条连接
   -- 非 CICS 程序使用这条连接来多次调用 CICS 程序
   -- 在完成调用后关闭并释放这个连接
2. EXEC CICS 接口，可以：
   -- 一次执行一个单独的调用（LINK PROGRAM）操作

第二章 一个简单的 CICS 程序

- CICS 程序框架
- 几个基本 CICS 命令
- 一个简单的 CICS 程序的源码
- 程序预编译、编译与连接
- 程序的配置与执行
- 程序的调试
- 练习

2-1 CICS 程序框架

- CICS 中 COBOL 语言程序的结构
- 嵌入式 CICS 命令
- EIB 和 COMMEREA

2-1-1 CICS 中 COBOL 语言程序的结构

IDENTIFICATION　DIVISION.
PROGRAM-ID.

~~ENVIRONMENT DIVTSION.~~
~~CONFIGURATION SECTION.~~
~~INPUT-OUTPUT SECTION~~

DATA DIVISION.
~~FILE SECTION.~~

WORKING-STORAGE SECTION.
LINKAGE SECTION.
PROCEDURE DIVISION.

EXEC CICS　……　END-EXEC.

EXEC CICS RETURN END-EXEC.
GOBACK.

以上是 CICS 中运行的 COBOL 程序的基本框架结构。为了保证资源的可恢复性和一致性，程序的所有 I/O 操作都要通过 CICS 接口来实现，所以 COBOL 中的 I/O 机制在这里是不能使用的，相应地 COBOL 程序里的环境部和其它文件 I/O 的定义是不能在 CICS 程序中出现的。同样，以下的 I/O 语句也不能在 PROCEDURE DIVISION 里出现。

ACCEPT
READ
OPEN
CLOSE
REWRITE
DISPLAY

这些命令完成的功能，要由相应的 CICS 接口命令来实现。
CICS 程序中使用 EXEC CICS RETURN END-EXEC.语句来终止程序，把控制返回给 CICS 系统。为了在 COBOL 编译过程中不出现编译警告或错误，在后边增加一条 GOBOACK 语句。

2-1-2 嵌入式 CICS 命令

EXEC CICS	SEND	FROM	（MY-DATA）	LENGTH	（40）	END-EXEC.
关键字	功能	选项	参数	选项	参数	结束标志

嵌入式 CICS 命令使用在 PROCEDURE DIVISION 里，每条命令由关键字、分隔符、功能名称、选项、参数和结束标志组成。命令的参数是 COBOL 程序里定义的数据变量，用来与 CICS 进行数据交换。CICS 提供适合多种语言的数据接口，所以对于参数类型的描述通常是以通用方式描述的，具体各种类型与实际 COBOL 变量对应如下：

数值（data-value）型变量：

数值型变量可以使用正确格式的常数或符合下面格式的COBOL变量来替换：

Halfword binary	—	PIC S9(4) COMP
Fullword binary	—	PIC S9(8) COMP
Doubleword unsigned binary	—	PIC 9(18) COMP
Character string	—	PIC X(n) 这里 n 是字符串的字节数

数据区（data-area）型变量：

只能使用符合格式的COBOL变量来替换，格式对应规则同上。

名字（name）型变量：

名字型变量可以使用引号引起的字符串常量或长度合适的COBOL字符串变量替换。

“hhmmss”格式变量：

这种变量一般用来表示时间，在COBOL语言中对应的格式是 PIC S9(7) COMP-3，这种变量的内容是“0HHMMSS+”格式，HH代表小时，从00到99，MM代表分钟，从00到59，SS代表秒，从00到59。

通用选项：

LENGTH：

大部分 CICS 命令都带有 LENGTH 选项，它带的参数指明数据参数的长度，是个 signed halfword binary 类型的变量，在 COBOL 中使用 S9(4) COMP 型变量。

RESP：

所有 CICS 命令都提供一个 RESP 选项，它的参数用来返回命令结果，COBOL 中使用 S9(8) COMP 型变量作参数，CICS 根据命令处理结果，把一个二进制数返回在这个变量里，一般与 DFHRESP( )宏比较以确定结果含义，如若与 DFHRESP(NORMAL)相等则说名处理成功，若与 DFHRESP(LENGERR)相等，则说明出现长度错误等等。

RESP2：提供比 RESP 更加详细的错误信息。

- EIB（EXEC interface block）
- COMMEREA（communication area）

EIB 和 COMMEREA 是 CICS 程序一般都要用到的数据区，在 CICS 预编译时，这两个数据区的定义会自动地添加到 COBOL 程序的 LINKAGE SECTION 里面，应用程序中可以直接引用。

CICS 系统为每个 TASK 建立一个 EIB 数据，里面记载这个 TASK 所在的系统环境情况，CICS 程序可以读取这些变量但是不能改变他们。

EIB 数据格式和内容如下：

```
01 DFHEIBLK.
  02 EIBTIME     PIC S9(7) COMP-3.    0HHMMSS 格式的时间
  02 EIBDATE     PIC S9(7) COMP-3.    0CYYDDD 格式的日期,C 代表世纪(0=1900,
                                      1=2000...),YY 是年,DDD 是从年初算的日数
  02 EIBTRNID    PIC X(4).            交易代码
  02 EIBTASKN    PIC S9(7) COMP-3.    任务号
  02 EIBTRMID    PIC X(4).            终端号
  02 DFHEIGDI    PIC S9(4) COMP.      保留
  02 EIBCPOSN    PIC S9(4) COMP.      光标位置
  02 EIBCALEN    PIC S9(4) COMP.      COMMAREA 的长度
  02 EIBAID      PIC X(1).            提醒（ATTENTION）标志
  02 EIBFN       PICTURE X(2).        最近执行过的 CICS 命令代码
  02 EIBRCODE    PIC X(6).            最近执行过的 CICS 命令返回信息
  02 EIBDS       PIC X(8).            最近使用过的文件名
  02 EIBREQID    PIC X(8).            最近使用过的 REQ ID
  02 EIBRSRCE    PIC X(8).            最近使用过的资源名
  02 EIBSYNC     PIC X.               标志程序结束前应当发送 SYNC 标志
  02 EIBFREE     PIC X.               资源释放标志
  02 EIBRECV     PIC X.               有数据可以 RECEIVE
  02 EIBSEND     PIC X.               SEND 标志
  02 EIBATT      PIC X.               接收数据中有 attach header 标志
  02 EIBEOC      PIC X.               接收数据中有 EOC 标志
```

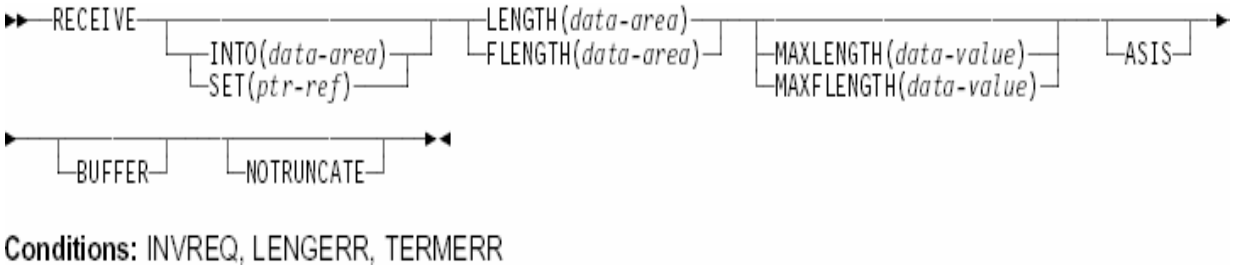
02 EIBFMH	PIC X.	接收数据 FMH 标志
02 EIBCOMPL	PIC X(1).	RECEIVE 终端的命令,输入数据是否结束
02 EIBSIG	PIC X(1).	接收数据中包含信号
02 EIBCONF	PIC X(1).	标志 APPC 会话中收到了 COMFIRM 请求
02 EIBERR	PIC X(1).	标志 APPC 会话中出现错误
02 EIBERRCD	PIC X(4).	APPC 错误码
02 EIBSYNRB	PIC X.	程序结束前应当发送 ROLLBACK，仅对 APPC 和 MRO
02 EIBNODAT	PIC X.	对方尚未发送信息
02 EIBRESP	PIC S9(8) COMP.	CICS 状态码
02 EIBRESP2	PIC S9(8) COMP.	状态码详细信息
02 EIBRLDBK	PIC X(1).	ROLLBACK 发生标志

COMMEREA 是不同 CICS 程序间或 CICS 程序与 CICS CLIENT 程序间传递信息的数据区，他的大小格式有传递此 COMMEREA 的两个程序自己约定。CICS 只负责把 COMMEREA 在不同程序间传递。

2-2 几个基本 CICS 命令

- RECEIVE 命令
- SEND 命令
- SEND CONTROL 命令
- RETURN 命令

2-2-1 RECEIVE 命令



Conditions: INVREQ, LENGERR, TERMERR

```
WORKING-STORAGE    SECTION.
01    XXX              PIC S9(8)    COMP.
01    YYY              PIC S9(8)    COMP.
01    RCV-BUF          PIC X(10).
01    RCV-LEN          PIC S9(4)    COMP VALUE 10.

PROCEDURE DIVISION.
      。。。
      EXEC CICS RECEIVE INTO(RCV-BUF)
            LENGTH(RCV-LEN)
            RESP(XXX) RESP2(YYY)
      END-EXEC.
      IF XXX NOT = DFHRESP(NORMAL)
            。。。
```

命令功能：RECEIVE 命令有多种格式，可以从多种输入渠道接收输入数据，这里给出的是从 3270 终端接收数据的命令格式。

选项：

INTO(data-area)：　指明接收数据区。

SET(ptr-ref)：　使用 ptr-ref 类型参量，返回输入数据指针。

LENGTH(data-area)：　Halfword binary value 类型参数，指明接受数据长度。

FLENGTH(data-area)：同 LENGTH 但是 fullword 型参量。

MAXLENGTH(data-value):Halfword binary value 型参量,指明可接受的最大长度。

MAXFLENGTH(data-value):MAXLENGTH 的 Fullword 版。

ASIS：　指明不把输入的小写字母自动翻译成大写字母

BUFFER：　指明按照 3270BUFFER 的格式读取数据。

NOTRUNCATE：　指明实际输入数据比所读取长的部分并不丢弃，可由下调 RECEVIE 命令读出

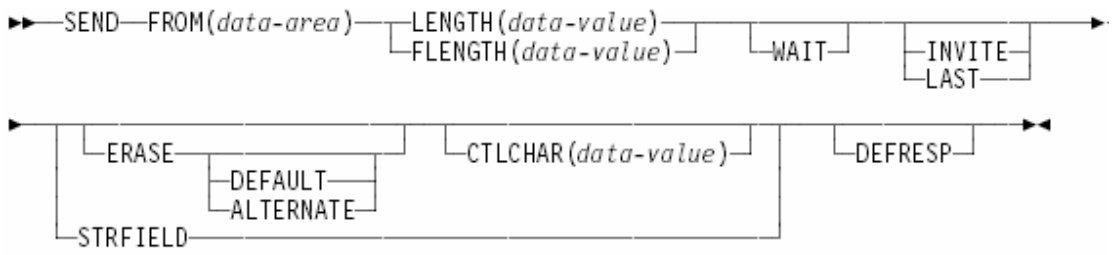
异常：

INVREQ： 非法请求，详细内容参考 RESP2。

LENGERR：实际输入数据长度超过 MAXLENGTH，并且 NOTRUNCATE 没有选用，长出部分数据已被丢弃

TERMERR： 相关终端发生错误。

2-2-2 SEND 命令



Conditions: INVREQ, LENGERR, TERMERR

```
DATA DIVISION.
01 WK-INFO                PIC    X(40).
01 RESP-CODE              PIC    S9(8) COMP.

PROCEDURE DIVISION.
  。。。
EXEC CICS SEND FROM(WK-INFO)
                   LENGTH(40)
                   RESP(RESP-CODE)
END-EXEC.
  。。。
```

命令功能：SEND 命令有多种格式，可以向多种输出渠道输出数据，这里给出的是向 3270 终端发送数据的命令格式。

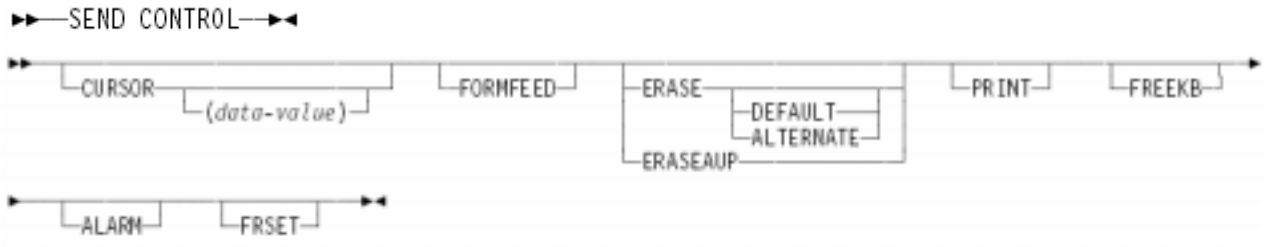
选项：

FROM(data-area)：	指明要发送数据的数据区
LENGTH(data-area)：	Halfword binary value 类型参数，指明发送数据长度。
FLENGTH(data-area)：	同 LENGTH 但是 fullword 型参量。
WAIT：	指明必须等待输出结束才能进行后续处理。
INVITE：	指明下调终端控制语句将是 RECEIVE 语句。
LAST：	指明这是本事务的最后一条 SEND 语句。
ERASE：	指明显示输出前先清屏，光标回到屏幕左上角。
DEFAULT：	指明终端使用默认屏幕大小。
ALTERNATE：	指明终端使用备用屏幕大小。
CTLCHAR(data-area)：	一个字节的 3270 终端控制字符,用来控制这次终端输出。
STRFIELD：	指明 FROM 的数据区里存放的是包含控制信息的 3270 输出结构，本选项生效时，ERASE 和 CTLCHAR 选项自动失效。
DEFRESP：	指明处理结束时要有明确的结果返回。

异常：

INVREQ： 非法请求，详细内容参考 RESP2。
LENGERR：实际输出数据长度选项参数值超出范围
TERMERR： 相关终端发生错误。

2-2-3 SEND CONTROL 命令



EXEC CICS SEND CONTROL CURSOR(1760) END-EXEC.

功能：向设备发送控制信息，上面所列是发送终端控制信息的格式。

选项：

CURSOR(data-value)：设置光标位置。3270 的标准终端是 24*80 个显示字符的设备，光标的位置用一个整数表示，第一行的位置是从 0 到 79，第二行是从 80 到 159，如此类推。

FORMFEED：指明先发送 FORMFEED 信号。

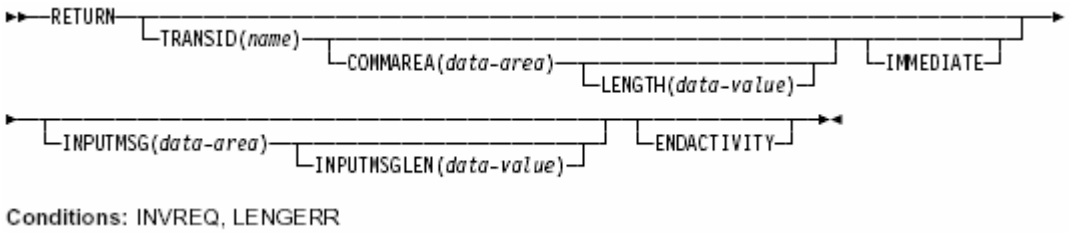
ERASE：清屏。

PRINT：启动 3270 打印机。

FREEKB：终端解锁。

ALARM：3270 终端发报警声。

2-2-4 RETURN 命令




```
DATA DIVISION.
01 WK-TRANSID              PIC    X(4).
01 WK-COMMAREA             PIC    X(100).

PROCEDURE DIVISION.
 。。。
 MOVE ‘TRN2’   TO   WK-TRANSID.
 MOVE    …     TO   WK-COMMAREA.
 EXEC CICS RETURN TRANSID(WK-TRANSID)
                  COMMAREA(WK-COMMAREA)
 END-EXEC.
```

功能：RETURN 命令把控制传给 CICS 系统或另一个交易。

选项：

TRANSID(name)：指明一个 1 到 4 字符的交易码。下一次输入信息从与发出 RETURN 命令的程序相关联的终端传来时，系统回使用这个交易来进行处理。

COMMAREA(data-area)：指明一段数据区，下一个接到控制的交易能够在它的 COMMAREA 里接收到这些数据。COMMAREA 允许传递数据区的最大长度是 32763 字节。

LENGTH(data-value)：指明上面所给的 COMMAREA 的长度（字节数）。

IMMEDIATE：指明下个交易立刻启动，并不等待相关终端的输入。

INPUTMSG(data-area)：指明一个可以传给下一个交易的数据区，下一个交易可以用 RECEIVE 命令得到这些数据。

INPUTMSGLEN(data-value)：给出上面的 INPUTMSG 长度，halfword 二进制参数。

ENDACTIVITY：BTS 专用参数。

异常：

LENGERR：根据 RESP2 的值区分，

11 代表 COMMAREA 的长度小于 0 或大于 32763；

26 代表 COMMAREA 区的地址是 0，但给出的长度不是 0；

27 代表 INPUTMSG 长度小于 0 或大于 32767.

INVREQ：根据 RESP2 的值区分，

1 代表 RETURN 带有 TRANSID 参数，但它没有关联的终端。

2 代表发送有 COMMAREA 或 IMMEDIATE 选项的 RETURN 命令的程序它并未处在最高的逻辑等级。

4 代表发送有 TRANSID 选项的 RETURN 命令另的程序，是在一个 APPC 的 LU 里。

8 代表发送有 INPUTMSG 选项的 RETURN 命令的程序没有相关联的终端。

200 代表发送有 INPUTMSG 选项的 RETURN 命令的程序，他本身是被 DPL 调起来的。

2-3 一个简单的 CICS 程序的源码

- 接收输入
- 进行处理
- 显示输出

2-3-1 完整的程序代码

```
IDENTIFICATION  DIVISION.
PROGRAM-ID SAMPLE1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  RESP-CODE    PIC S9(8)  COMP.
01  RCV-BUF      PIC X(80).
01  RCV-LEN      PIC S9(4)  COMP VALUE 10.
01  WK-INFO      PIC X(40).
LINKAGE SECTION.
PROCEDURE DIVISION.
    EXEC CICS RECEIVE INTO(RCV-BUF)
         LENGTH(RCV-LEN)
         MAXLENGTH(80)
         RESP(RESP-CODE)
    END-EXEC.
    IF RESP-CODE NOT = DFHRESP(NORMAL)
       PERFORM  SHOW-CICS-ERROR
       EXEC CICS RETURN END-EXEC
    END-IF.
    MOVE 'HELLO,WORLD' TO WK-INFO.
    EXEC CICS SEND FROM(WK-INFO)
               LENGTH(12)
               ERASE
               RESP(RESP-CODE)
    END-EXEC.
    EXEC  CICS  SEND  CONTROL  CURSOR(160)  END-EXEC.
    STRING 'YOUR TRANSACTION ID:' EIBTRNID DELIMITED BY SIZE
           INTO WK-INFO.
    EXEC CICS SEND FROM(WK-INFO)
               LENGTH(40)
               RESP(RESP-CODE)
    END-EXEC.
    EXEC  CICS  SEND  CONTROL  CURSOR(320)  END-EXEC.
    MOVE SPACES TO WK-INFO.
    STRING 'YOUR INPUT ARE:' RCV-BUF DELIMITED BY SIZE
           INTO WK-INFO.
    EXEC CICS SEND FROM(WK-INFO)
               LENGTH(40)
               RESP(RESP-CODE)
    END-EXEC.
    EXEC CICS RETURN END-EXEC.
    GOBACK.
```

2-3-1 完整的程序代码（续页）

```
SHOW-CICS-ERROR.
    MOVE ’CICS ERROR’ TO WK-INFO.
    EXEC CICS SEND FROM(WK-INFO)
              LENGTH(40)
              RESP(RESP-CODE)
    END-EXEC.
SHOW-CICS-ERROR-END.
    EXIT.
```

程序说明：

程序首先读取终端输入命令，根据返回码检查是否成功，若取输入成功，清屏，输出一行“HELLO，WORLD”，然后把光标移动到第 3 行从 EIB 信息中输出交易码，然后把光标移动到第 5 行，显示前面终端输入的所有信息。最后程序把控制返回返回给 CICS。

2-4 程序预编译、编译与连接

- 处理用的 JCL 过程
- 处理的 JCL 语句

2-4-1 处理的 JCL 过程

```
//DFHYITVL PROC SUFFIX=1$,
//         INDEX='CICSTS13.CICS',                  CICS 数据集的前缀
//         PROGLIB='EDU.ONLINE.LOADLIB',           连接后的执行码存放数据集
//         MEMB='GO'                               编译生成的成员名
//         DSCTLIB='CICSTS13.CICS.SDFHCOB',        CICS COBOL 库
//         AD370HLQ='IGY',                         COBOL 数据集前缀
//         LE370HLQ='CEE',                         连接函数库前缀
//         OUTC=A,
//         REG=4M,
//         LNKPARM='LIST,XREF',
//         STUB='DFHEILID',
//         LIB='SDFHC370',
//         WORK=SYSDA
//TRN      EXEC PGM=DFHECP&SUFFIX,                 预编译步骤
//              PARM='COBOL3',
//              REGION=&REG
//STEPLIB  DD DSN=&INDEX..SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC                         预编译信息
//SYSPUNCH DD DSN=&&SYSCIN,                        预编译生成的 COBOL 源码
//              DISP=(,PASS),UNIT=&WORK,
//              DCB=BLKSIZE=400,
//              SPACE=(400,(400,100))
//*
//COB      EXEC PGM=IGYCRCTL,REGION=&REG,          COBOL 编译步骤
//         PARM='NODYNAM,LIB,OBJECT,RENT,RES,APOST,MAP,XREF'
//STEPLIB  DD DSN=&AD370HLQ..SIGYCOMP,DISP=SHR
//SYSLIB   DD DSN=&DSCTLIB,DISP=SHR                    编译使用到的库
//           DD DSN=&INDEX..SDFHCOB,DISP=SHR
//           DD DSN=&INDEX..SDFHMAC,DISP=SHR
//           DD DSN=&INDEX..SDFHSAMP,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC                             编译信息
//SYSIN    DD DSN=&&SYSCIN,DISP=(OLD,DELETE)           输入的源码
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),           输出目标码
//              UNIT=&WORK,SPACE=(80,(250,100))
//SYSUT1   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT2   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT3   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT4   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT6   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT7   DD UNIT=&WORK,SPACE=(460,(350,100))
```

```
//*
//COPYLINK EXEC PGM=IEBGENER,COND=(7,LT,COB)          拷贝 CICS 存根代码步骤
//SYSUT1   DD DSN=&INDEX..&LIB(&STUB),DISP=SHR       存根代码原文件
//SYSUT2   DD DSN=&&COPYLINK,DISP=(NEW,PASS),        存根代码临时文件
//            DCB=(LRECL=80,BLKSIZE=400,RECFM=FB),
//            UNIT=&WORK,SPACE=(400,(20,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DUMMY
//*
//LKED   EXEC PGM=IEWL,REGION=&REG,                  连接步骤
//            PARM='&LNKPARM',COND=(5,LT,COB)
//SYSLIB   DD DSN=&INDEX..SDFHLOAD,DISP=SHR
//         DD DSN=&LE370HLQ..SCEELKED,DISP=SHR
//SYSLMOD  DD DSN=&PROGLIB(&MEMB),DISP=SHR           输出可执行码
//SYSUT1   DD UNIT=&WORK,DCB=BLKSIZE=1024,
//            SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSLIN   DD DSN=&&COPYLINK,DISP=(OLD,DELETE)       输入程序目标码
//         DD DSN=&&LOADSET,DISP=(OLD,DELETE)        输入存根代码临时文件
//         DD DDNAME=SYSIN
```

CICS 提供一个编译联机 COBOL 程序 JCL 过程，一般放在 CICSTS13.CICS.SDFHPROC（DFHYITVL），这可能根据 CICS 安装的情况而不同。通常可以把这个过程拷贝到自己的 JCL 过程目录中去，然后按照自己系统中各种软件安装配置情况进行修改，作为自己的编译过程。

2-4-2 处理的 JCL 语句

```
//COMP      JOB NOTIFY=&SYSUID
//MYPROC    JCLLIB ORDER='DEVP01.JCL'
//TRCPLNK   EXEC DFHYITVL,MEMB=SAMPLE1
//TRN.SYSIN   DD DSN=DEVP01.SRC(SAMPLE1),DISP=SHR
```

编辑如上的 JCL 文件，用来处理生成程序的执行代码。

```
//MYPROC    JCLLIB ORDER='DEVP01.JCL'：
```
指向自己的 JCL 过程所在库

```
//TRCPLNK   EXEC DFHYITVL,MEMB=SAMPLE1
```
调用上面修改过的 DFHYITVL 过程，MEMB 给出生成执行码成员名

```
//TRN.SYSIN   DD DSN=DEVP124.JCL(SAMPLE1),DISP=SHR
```
覆盖 TRN.SYSIN，指向源程序

提交 JCL，使用 SDSF 去查看处理结果，根据结果改正程序语法错误。

2-5 程序的配置执行

- 需要的 CICS 配置
- 交易的执行

2-5-1 需要的 CICS 配置

- 定义程序
- 定义交易
- 安装程序
- 安装交易

CICS 程序要定义并安装到 CICS 系统中才能够运行，这里先简要介绍以下定义和安装的命令，详细情况参见后面的专门介绍。

CICS 中通过 CEDA 交易可以联机进行资源定义：
开启 CICS 终端，执行用户登陆后依次执行以下交易

```
CEDA DEF PROG(SAMPLE1) GROUP(TESTGP)
CEDA DEF TRANS(TRN1) GROUP(TESTGP) PROGRAM(SAMPLE1)
CEDA INS PROG(SAMPLE1) GROUP(TESTGP)
CEDA INS TRANS(TRN1) GROUP(TESTGP)
```

安装好的程序，可以使用 CEMT INQUIRE PROG(???)进行查询，交易可以用 CEMT INQUIRE TRANSACTION（？？？）进行查询。

2-5-2 程序的运行

输入： TRN1　AAB

输出：

```
HELLO,WORLD

YOUR TRANSACTION ID:TRN1

YOUR INPUT ARE:TRN1 AAB
```

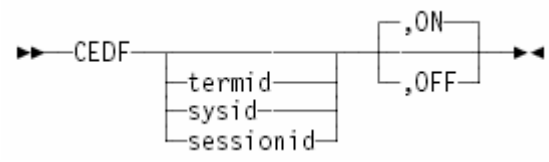
CICS 程序的运行，只要在 CICS 终端上直接输入 TRANSACTION ID，CICS 系统会自动到配置信息中去查找这个交易的定义，找到它的第一只程序装载近来运行。上面的画面就是交易输入的命令和处理输出的结果。

2-6 CICS 程序的调试

CICS 提供一个系统联机事务 CEDF(Commad Execution Diagnostic Facility)，可以用来跟踪一个 CICS 程序中执行的所有 CICS 命令。

- CEDF 的启动
- TASK 环境中的 EIB 信息
- CICS 命令的拦截
- 显示和修改工作存储空间
- 程序中增加追踪入口
- 限定追踪特定的 CICS 命令
- 追踪伪会话交易
- 异常终止程序

2-6-1 CEDF 的启动



CEDF 能够在程序初始化时、程序中每条 CICS 命令运行前和程序结束时截取到程序的状态。CEDF 能帮助你查找到程序中的错误。

CEDF 的选项：

sessionid：

指明一个 SESSION ID,被测试的程序通过这个 SESSION(MRO, APPC,或 LU6.1)被调用。凡是使用这个 SESSION 的交易都会被 CEDF 拦截。

sysid：

指明一个远程 CICS 系统的系统 ID，凡是从这个系统连接过来的交易都会被 CEDF 拦截。

termid:

指明一个终端 ID,凡是从这个终端发起的交易都会被 CEDF 拦截.如果没有输入终端号，默认使用当前终端。

OFF:

调试过程结束。输入 OFF 选项时前面必须输入一个逗号，如 “CEDF ，OFF”或“CEDF FC3A，OFF”。

ON：

调试过程开始。省缺时，ON 是默认选项，“CEDF 0012”和“CEDF 0012，ON”是等价的。

2-6-2 TASK 环境中的 EIB 信息

```
 TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010142 APPLID: DVPCICS1 DISPLAY:  00
   STATUS:  PROGRAM INITIATION

      EIBTIME      = 143827
      EIBDATE      = 0104211
      EIBTRNID     = 'TRN1'
      EIBTASKN     = 10142
      EIBTRMID     = '0074'

      EIBCPOSN     = 4
      EIBCALEN     = 0
      EIBAID       = X'7D'                                    AT X'002150EA'
      EIBFN        = X'0000'                                  AT X'002150EB'
      EIBRCODE     = X'000000000000'                          AT X'002150ED'
      EIBDS        = '........'
+     EIBREQID     = '........'



 ENTER:  CONTINUE
 PF1 : UNDEFINED          PF2 : SWITCH HEX/CHAR     PF3 : END EDF SESSION
 PF4 : SUPPRESS DISPLAYS  PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : SCROLL BACK        PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY   PF11: EIB DISPLAY         PF12: UNDEFINED
```

上面是跟踪交易 TRN1 的初始化画面，这里显示这个 TASK 的 EIB 信息。画面第一行显示系统的当前信息，TRANSACTION: 交易代码 PROGRAM: 程序名 TASK:任务编号 APPLID:CICS 系统的程序 ID; 画面第二行显示当前程序的状态; 画面中间显示的是本 TASK 的当前 EIB 信息，部分变量右侧显示的“AT X'002150EB'”字样是给出这些变量实际的地址信息。最下面显示的是当前界面的功能键定义，这些键的定义在整个 CEDF 中是比较一致的：

PF2：显示数据在字符方式和二进制方式间切换

PF3：结束调试过程

PF4: 跳过显示（具体用途见后面介绍）

PF5：进入内存显示/编辑状态

PF6：显示用户输出画面

PF7：向上翻屏

PF8：向下翻屏

PF9：指明停止条件

PF10: 回现上个跟踪画面

PF11: 显示 EIB 信息

PF12: 初始化界面里没有定义，后面界面用来 ABEND 被跟踪程序

可以使用 PF7 和 PF8 功能键来上下翻动显示全部 EBI 信息。提交键用来使 CEDF 进入到下一条 CICS 命令画面。程序执行中可以按 PF11 键来随时查看 EIB 信息。

2-6-3 CICS 命令的拦截

画面 1

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010305 APPLID: DVPCICS1 DISPLAY:  00
  STATUS:  ABOUT TO EXECUTE COMMAND
  EXEC CICS SEND
   FROM ('HELLO,WORLD ')
   LENGTH (12)
   ERASE
   NOHANDLE




  OFFSET:X'000584'     LINE:00026          EIBFN=X'0404'


 ENTER:  CONTINUE
 PF1 : UNDEFINED           PF2 : SWITCH HEX/CHAR     PF3 : UNDEFINED
 PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : SCROLL BACK         PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY    PF11: EIB DISPLAY         PF12: ABEND USER TASK
```

画面 2：

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010305 APPLID: DVPCICS1 DISPLAY:  00
  STATUS:  COMMAND EXECUTION COMPLETE
  EXEC CICS SEND
   FROM ('HELLO,WORLD ')
   LENGTH (12)
   ERASE
   NOHANDLE




  OFFSET:X'000584'     LINE:00026          EIBFN=X'0404'
  RESPONSE: NORMAL                         EIBRESP=0

 ENTER:  CONTINUE
 PF1 : UNDEFINED           PF2 : SWITCH HEX/CHAR     PF3 : END EDF SESSION
 PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : SCROLL BACK         PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY    PF11: EIB DISPLAY         PF12: ABEND USER TASK
```

CEDF 会在每条 CICS 命令执行前把程序拦截下来，显示这条命令的相关信息，用户可以在这时察看程序执行情况或者对一些变量或内存进行修改，然后执行它。

上面画面 1 显示的就是 SEND 命令执行前的程序情况，CEDF 显示命令所选择的选项和相关参数的数值。程序员可以在这里对 CICS 命令的参数进行修改然后再执行。按输入键，CEDF 将在命令执行完后再次把程序拦截下来，这里可以看到命令的执行情况和执行完后参量的变化。

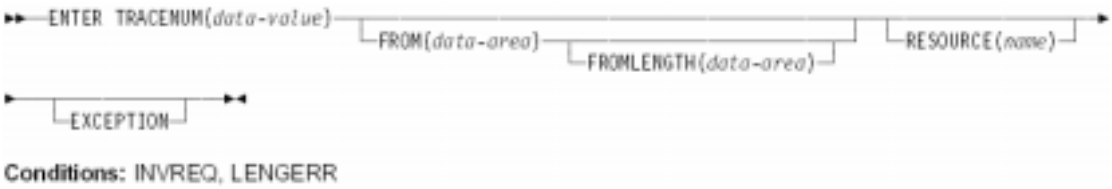
2-6-4 显示和修改工作存储空间

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010327 APPLID: DVPCICS1 DISPLAY:  00
  STATUS:  ABOUT TO EXECUTE COMMAND
  EXEC CICS SEND
   FROM (X'C8C5D3D3D66BE6D6D9D3C440')                              AT X'1AB08498'
   LENGTH (X'000C')                                                AT X'001E6D30'
   ERASE
   NOHANDLE



  OFFSET:X'000584'     LINE:00026          EIBFN=X'0404'


 ENTER:  CONTINUE
 PF1 : UNDEFINED           PF2 : SWITCH HEX/CHAR     PF3 : UNDEFINED
 PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : SCROLL BACK         PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY    PF11: EIB DISPLAY         PF12: ABEND USER TASK
```

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010327 APPLID: DVPCICS1 DISPLAY:  00
  ADDRESS: 1AB08498                    WORKING STORAGE+X'000068'
 1AB08490   000000                     C8C5D3D3 D66BE6D6   ........HELLO,WO
 1AB084A0   000008   D9D3C440 40404040 40404040 40404040   RLD
 1AB084B0   000018   40404040 40404040 40404040 40404040
 1AB084C0   000028   000C0000 7FFFFFFF 3C100000 00800000   ....″...........
 1AB084D0   000038   7F7FFFFF 34000000 7F800000 FF800000   ″″.....″........
 1AB084E0   000048   7FA00000 FFA00000 7FC00000 FFC00000   ″.......″{...{..
 1AB084F0   000058   00000010 0006000E 00000000 00000000   ................
 1AB08500   000068   00000000 00000000 FFB2FFB2 003F003F   ................
 1AB08510   000078   003F004B 004B004B 00010002 00180035   ................
 1AB08520   000088   00710006 000F0021 FF83FC03 C003FFDB   .........c..{...
 1AB08530   000098   FECDECBD 00800400 40000026 01341344   ........ .......
 1AB08540   0000A8   03330417 00000000 00000000 00000000   ................
 1AB08550   0000B8   00000000 1AB09418 00000000 00000000   ......m.........
 1AB08560   0000C8   00000000 00000000 00000000 00000000   ................
 1AB08570   0000D8   00000000 00000000 00000000 00000000   ................
 1AB08580   0000E8   C7C5D5C5 00000000 00000000 00000000   GENE............

 ENTER:  CURRENT DISPLAY
 PF1 : UNDEFINED           PF2 : BROWSE TEMP STORAGE PF3 : UNDEFINED
 PF4 : EIB DISPLAY         PF5 : INVOKE CECI         PF6 : USER DISPLAY
 PF7 : SCROLL BACK HALF    PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
 PF10: SCROLL BACK FULL    PF11: SCROLL FORWARD FULL PF12: REMEMBER DISPLAY
```

CEDF 拦截到 CICS 命令时可以查看和修改当时的工作内存区。在拦截到的 CICS 命令界面按 PF2 键，CICS 参数显示切换到二进制状态，在每个参数同行的屏幕右侧，显示此参数所在的内存地址。按 PF5 键进入内存显示和修改状态。在第二行输入数据的内存地址（可在命令界面直接得到或根据编译时的输出得到），回车，就会显示指定内存的数据内容。这时如果需要修改，可以直接修改某一地址的数据，回车落实修改。结束后回车，回到原命令界面。

2-6-5　程序中增加追踪入口



EXEC CICS ENTER TRACENUM(1) FROM(WK-INFO) END-EXEC.

CEDF 只能拦截 CICS 命令，无法看到程序宿主语言中的变量与数据的变化情况，为此，CICS 提供一条命令 ENTER TRACENUM，把这条命令加再需要观察变量内容的任何逻辑点上，通过这条命令，可以把程序中任何变量在 CEDF 中显示出来。

选项：

TRACENUM(data-value)：	给出一个 TRACE 编号，用以标示一条 ENTER TRACENUM 语句。
FROM(data-area)：	指明需要查看的变量或数据区。
FROMLENGTH(data-area)：	指明一个 halfword 的二进制变量，给出要查看变量的长度。
RESOURCE(name)：	给出一个要查看的资源名。
EXCEPTION：	指示 CICS 写一条用户数据例外。

例外

INVREQ：非法请求，根据 RESP2 ：

1 TRACENUM 超出了 0-199 的范围

2 没有合法的 TRACE 目标

3 用户的 TRACE 标志是关闭的。

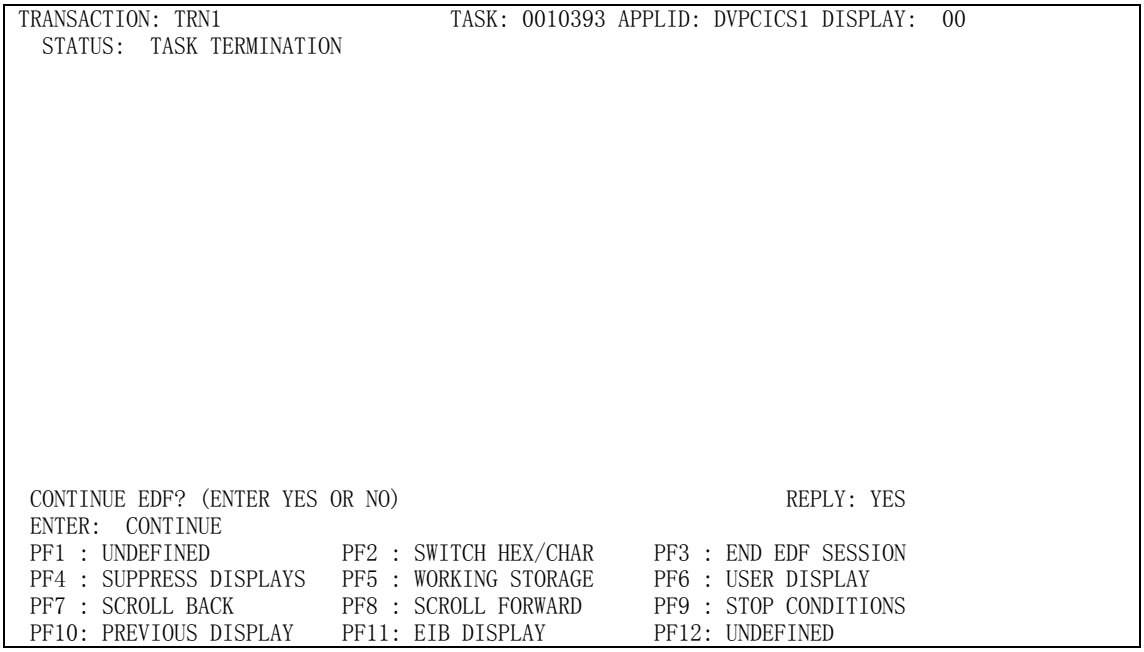
LENGERR：FROMLENGTH 超出了 0-4000 的合法范围。

2-6-6　限定追踪特定的 CICS 命令

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010327 APPLID: DVPCICS1 DISPLAY:  00
  DISPLAY ON CONDITION:-

       COMMAND:              EXEC CICS  SEND CONTROL
       OFFSET:                          X'......'
       LINE NUMBER:                     ........
       CICS EXCEPTIONAL CONDITION:      ERROR
       ANY CICS CONDITION               NO
       TRANSACTION ABEND                YES
       NORMAL TASK TERMINATION          YES
       ABNORMAL TASK TERMINATION        YES

       DLI ERROR STATUS:                ..
       ANY DLI ERROR STATUS




 ENTER:  CURRENT DISPLAY
 PF1 : UNDEFINED           PF2 : UNDEFINED           PF3 : UNDEFINED
 PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : UNDEFINED           PF8 : UNDEFINED           PF9 : UNDEFINED
 PF10: UNDEFINED           PF11: UNDEFINED           PF12: REMEMBER DISPLAY
```

如果一个程序很长，其中有很多 CICS 命令，调试时如果逐条追踪到后面的命令，是一件十分繁琐的事情。CEDF 中提供了一种手段，能够追踪某条特定的 CICS 命令。在 CEDF 中，按 PF9，就会出现上面的画面，在“COMMAND:　　EXEC CICS”后输入你想追踪的 CICS 命令名，然后按 PF4 键，CEDF 就会跳过中间其他 CICS 命令的显示，直接停到所要追踪的 CICS 命令上。再次按 PF4 将停在下一条要追踪的 CICS 命令上。

2-6-7 追踪伪会话交易



伪会话交易是通过若干个交易来完成一个会话，CEDF 提供一定的机制使用户能够连续地追踪一个伪对话过程中的全部交易。在 CEDF 追踪的一个交易结束后会显示如上画面，提问程序跟踪是否继续，若回答是 YES，那么符合跟踪条件的下次交易（伪会话过程中的下一个交易）提交上来时，CEDF 能够继续跟踪。

2-6-8 异常终止程序

```
TRANSACTION: TRN1 PROGRAM: SAMPLE1  TASK: 0010410 APPLID: DVPCICS1 DISPLAY:  00
  STATUS:  ABOUT TO EXECUTE COMMAND
  EXEC CICS SEND
   FROM ('HELLO,WORLD ')
   LENGTH (12)
   ERASE
   NOHANDLE









  OFFSET:X'000584'     LINE:00026          EIBFN=X'0404'

 ENTER ABEND CODE AND REQUEST ABEND AGAIN                        REPLY: AAAA
 ENTER:  CONTINUE
 PF1 : UNDEFINED           PF2 : SWITCH HEX/CHAR     PF3 : UNDEFINED
 PF4 : SUPPRESS DISPLAYS   PF5 : WORKING STORAGE     PF6 : USER DISPLAY
 PF7 : SCROLL BACK         PF8 : SCROLL FORWARD      PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY    PF11: EIB DISPLAY         PF12: ABEND USER TASK
```

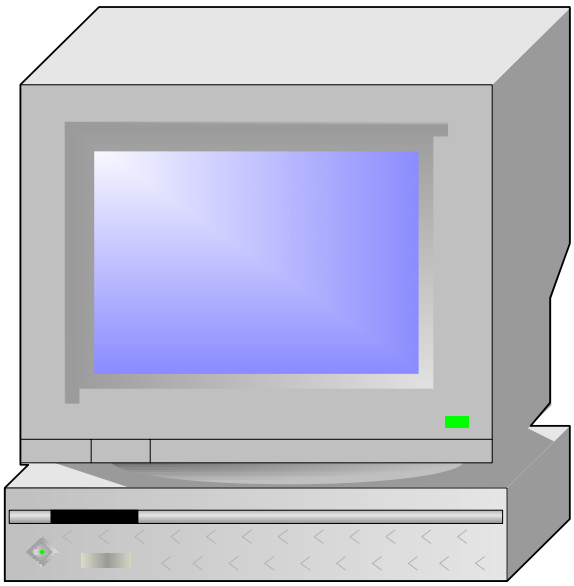
追踪一个比较长的程序时如果想中途停止，不再继续追踪，可以使程序 ABEND，停止执行。在 CEDF 中按 PF12 键，在当前画面中出现一行新的显示“ENTER ABEND CODE AND REQUEST ABEND AGAIN REPLY: ”要求输入一个 ABEND CODE ，输入后在按 PF12 键，发送 ABEND，程序非正常结束。

练习 1

第三章　基本映像支持（**BMS**）编程

- 3270 终端
- 屏幕映像（MAP）的定义
- 屏幕映像（MAP）的使用
- 使用 BMS 程序的编译与执行

3-1　3270 终端



IBM 3270 终端：

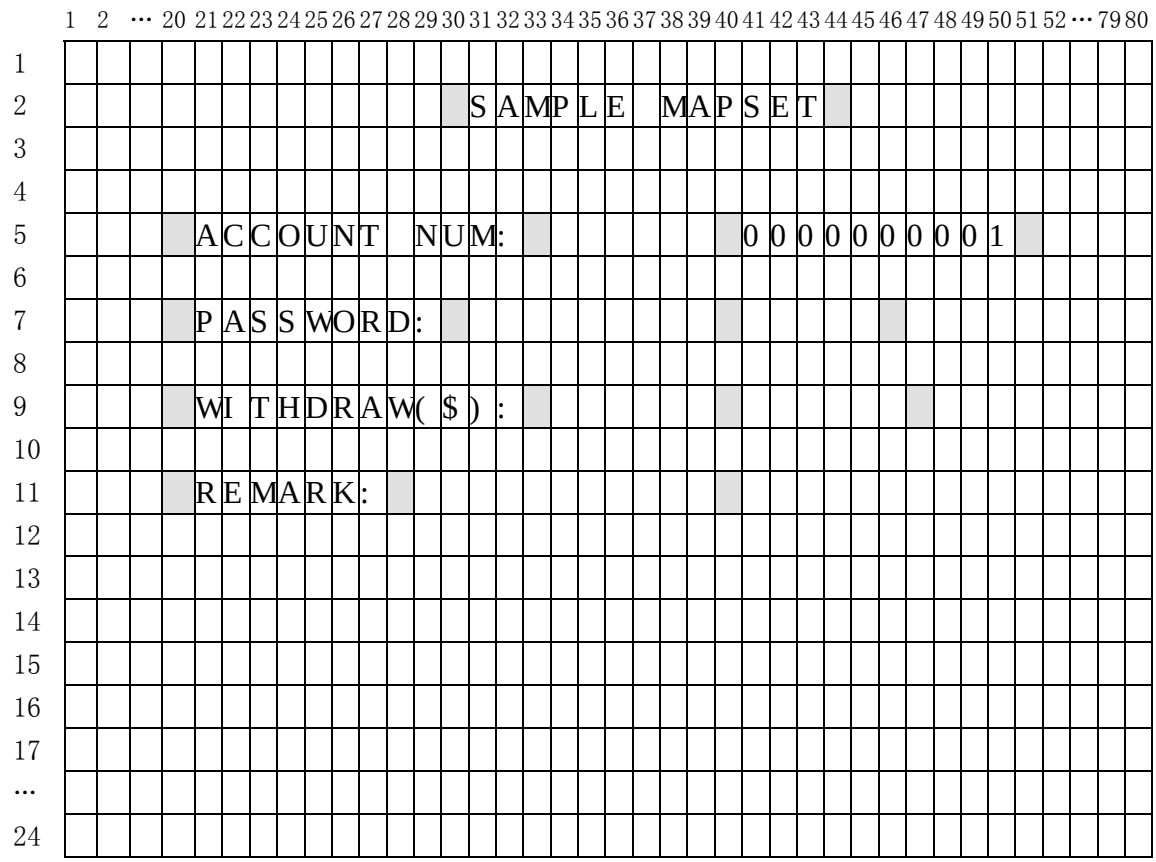
IBM 3270 是 IBM 制造的一系列终端的统称，这种设备最早起源于 IBM3270 系统。3270 终端使用一种叫做 3270 数据流（datastream）的技术来减少系统与用户交互时终端的发生，在这种数据流里，显示和控制在一起编码，每次数据输出的数据就能绘画整个屏幕。3270 终端使用了“域”的概念，这使得整个屏幕能够被划分为很多由多个连续的位置组成的小单元，每域能够被赋予特定的属性（如颜色、高亮、可以修改、不可修改等等）。同时使用一种叫做“读修改（Read Modified）”的技术，被修改过的域可以一次被系统读入

3270 终端已开始有 12 个，后来发展到 24 个功能键（Programmed Function Keys，PFK）。当任意一个功能键按下时，就会引发一个设备终断，能向主机发送信息表明是哪个键被按下了。程序一般把一些常用功能（如向上翻页、向下翻页、后退等）定义到某个功能键上，以方便用户操作。

3-2　屏幕映像（MAP）的定义

- 屏幕中的域
- 屏幕映像集
- DFHMSD 宏
- DFHMDI 宏
- DFHMDF 宏

3-2-1　屏幕中的域



屏幕上的域

标准屏幕划分为 24X80 个空位大小，域就是这个空间上一个占有固定的长度、位置，并有一定属性的区域。域的属性由属性子节定义，每个属性字节本身占用一个空格，它本身是不可见的，每个域的位置起始于一个属性字节，终止于下一个属性字节，其中的空位拥有前面属性字节规定的属性。

域的属性：

一个域通常可以有以下的属性：

PROTCTED/UNPORTECTED：表明终端上一个域中的数据能否接受修改。凡有 PROTCTED 特性的域不能被修改，UNPORTECTED 属性的域可以修改。。

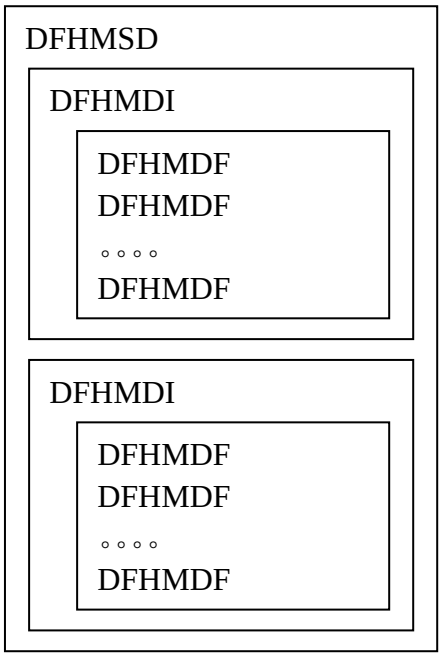
ASKIP：表明光标会自动从这个域跳过。终端上按 TAB 键使光标在各个域间切换时，会自动跳过有 ASKIP 属性的域，自动跳到下一个 UNPORTECTED 属性的域上。

NUM：表明此域只接受数字字符的输入。数字字符包括 0-9，正负号和小数点。

域的定义：

屏幕中的域是用汇编语言的宏定义的，例如如上的屏幕格式各个域的定义如下：

```
         DFHMDF POS=(2,30),ATTRB=(ASKIP,NORM),LENGTH=13,                X
                INITIAL='SAMPLE MAPSET'
         DFHMDF POS=(5,20),ATTRB=(ASKIP,NORM),LENGTH=12,                X
                INITIAL='ACCOUNT NUM:'
ACCT     DFHMDF POS=(5,40),ATTRB=(UNPORT,IC),LENGTH=10,OUTLINE=(UNDER)
         DFHMDF POS=(5,51),ATTRB=PROT,LENGTH=1
         DFHMDF POS=(7,20),ATTRB=(ASKIP,NORM),LENGTH=9,                 X
                INITIAL='PASSWORD:'
PWD      DFHMDF POS=(7,40),ATTRB=(UNPORT,DRK),LENGTH=6,OUTLINE=(UNDER)
         DFHMDF POS=(7,47),ATTRB=PROT,LENGTH=1
         DFHMDF POS=(9,20),ATTRB=(ASKIP,NORM),LENGTH=12,                X
                INITIAL='WITHDRAW(\):'
MONEY    DFHMDF POS=(9,40),ATTRB=(UNPORT,NUM),LENGTH=6,OUTLINE=(UNDER)
         DFHMDF POS=(9,47),ATTRB=PROT,LENGTH=1
         DFHMDF POS=(11,20),ATTRB=(ASKIP,NORM),LENGTH=7,                X
                INITIAL='REMARK:'
REMARK   DFHMDF POS=(11,40),ATTRB=(UNPORT,NORM),LENGTH=20,              X
                OUTLINE=(UNDER)
         DFHMDF POS=(11,61),ATTRB=PROT,LENGTH=1
```



屏幕上的域由 DFHMDF 宏来定义，每个域代表屏幕上一段连续的位置，若干个域一起，组成一个屏幕映像（MAP）。屏幕映像由 DFHMDI 宏来定义，每个屏幕映像对应一个特定格式的屏幕，若干个屏幕映像，组成一个屏幕映像集（MAPSET）。屏幕映像集由 DFHMSD 宏来定义，它包括特定程序使用的若干个屏幕映像。屏幕映像集是 CICS 的一种资源，CICS 程序调用 MAP 时必须指明它所在的 MAPSET。

包括前面所示的一个屏幕映像，和另一个输出用屏幕映像的完整的屏幕映像集定义如下：

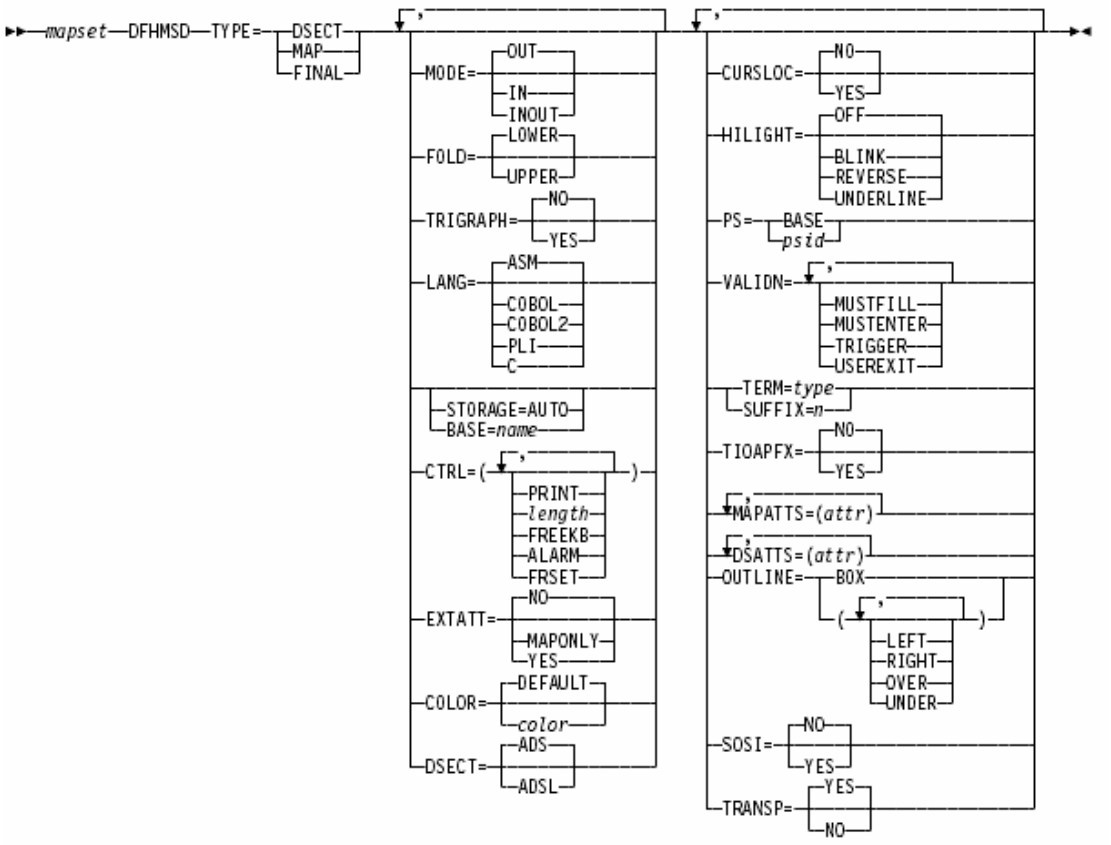
```
TSTMPS1 DFHMSD TYPE=MAP,MODE=INOUT,LANG=COBOL,                          X
               STORAGE=AUTO,TIOAPFX=YES
TSTMP1  DFHMDI SIZE=(24,80),CTRL=(PRINT,FREEKB),MAPATTS=(OUTLINE,SOSI)
        DFHMDF POS=(2,30),ATTRB=(ASKIP,NORM),LENGTH=13,                 X
               INITIAL='SAMPLE MAPSET'
        DFHMDF POS=(5,20),ATTRB=(ASKIP,NORM),LENGTH=12,                 X
               INITIAL='ACCOUNT NUM:'
ACCT    DFHMDF POS=(5,40),ATTRB=(UNPORT,IC),LENGTH=10,OUTLINE=(UNDER)
        DFHMDF POS=(5,51),ATTRB=PROT,LENGTH=1
        DFHMDF POS=(7,20),ATTRB=(ASKIP,NORM),LENGTH=9,                  X
               INITIAL='PASSWORD:'
```

```
PWD      DFHMDF POS=(7,40),ATTRB=(UNPORT,DRK),LENGTH=6,OUTLINE=(UNDER)
         DFHMDF POS=(7,47),ATTRB=PROT,LENGTH=1
         DFHMDF POS=(9,20),ATTRB=(ASKIP,NORM),LENGTH=12,                X
               INITIAL='WITHDRAW(\):'
MONEY    DFHMDF POS=(9,40),ATTRB=(UNPORT,NUM),LENGTH=6,OUTLINE=(UNDER)
         DFHMDF POS=(9,47),ATTRB=PROT,LENGTH=1
         DFHMDF POS=(11,20),ATTRB=(ASKIP,NORM),LENGTH=7,                X
               INITIAL='REMARK:'
REMARK   DFHMDF POS=(11,40),ATTRB=(UNPORT,NORM),LENGTH=20,              X
               OUTLINE=(UNDER)
         DFHMDF POS=(11,61),ATTRB=PROT,LENGTH=1
TSTMP2   DFHMDI SIZE=(24,80),CTRL=(PRINT,FREEKB),MAPATTS=(OUTLINE,SOSI)
         DFHMDF POS=(1,30),ATTRB=(ASKIP),LENGTH=20,                     X
               INITIAL='TRANSACTION FINISHED'
OUTMSG   DFHMDF POS=(3,20),ATTRB=(ASKIP),LENGTH=60
         DFHMSD TYPE=FINAL
         END
```

MAPSET 的定义使用的是汇编宏，在书写格式上必须遵守汇编语言的规定：

1) 标号起始于第一列
2) 汇编宏起始于第 10 列
3) 各种操作起始于第 16 列
4) 若须续行，则该行的第 72 列必须含有一个非空的“连续标志符”，表明下一行为须行。
5) 续行的操作码必须从 16 列开始

3-2-3　DFHMSD 宏



一些常用选项的说明：

TYPE：　指明要产生的是物理屏幕映像集（TYPE=MAP）还是符号屏幕映像集（TYPE=DSECT）。

MODE：　指明定义的屏幕映像集是用来把数据输出到终端（MODE= OUT），还是用来从终端读取数据（MODE=IN），或者是既用来输出又用来输入数据（MODE=INOUT）。

LANG：　生成符号屏幕映像集时才需要此参数，用来指定生成的符号屏幕映像集使用的语言，可以使 COBOL、PL/I、ASMEMBLER 或 C 语言。

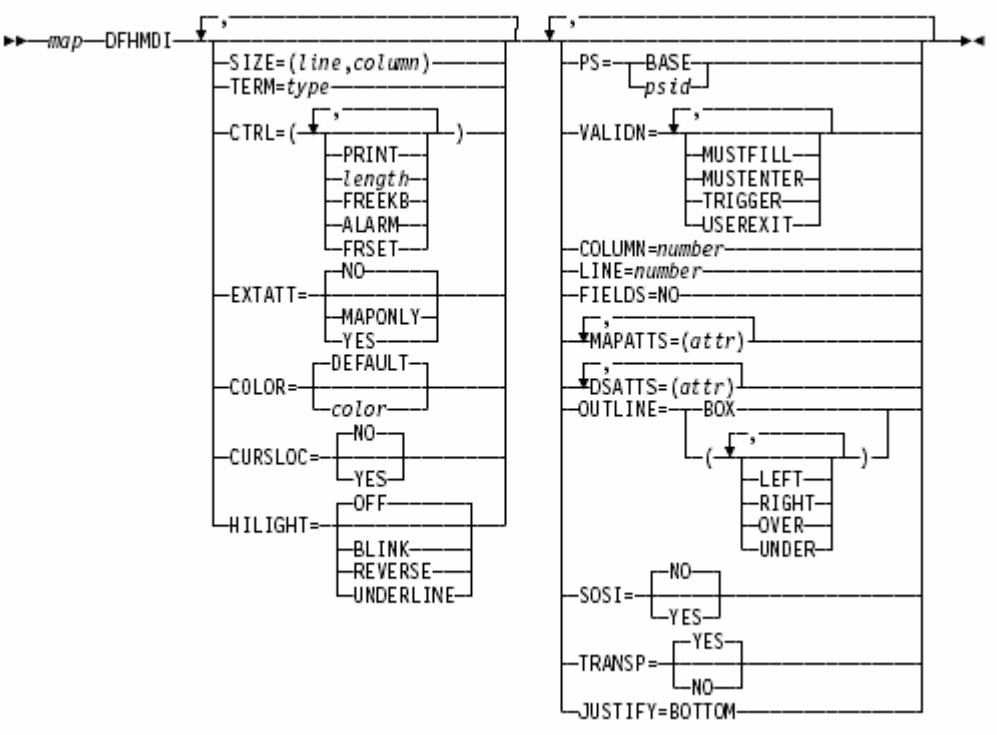
TERM：　指明使用屏幕映像集的终端的具体类型。如 TERM=3270-2，指明使用的是 IBM 3270 Model 2 终端设备。符号屏幕映像集不需要此选项。

TIOAPFX：　如果想在 CICS 命令中使用此屏幕映像集，必须设置 TIOAPFX=YES，否则将会因终端输入输入出流覆盖 TIOA（Terminal Input/Output Area）而造成程序失败。

STORAGE：　对 COBOL 程序而言，如果一个屏幕映像集中有多个屏幕映像，则必须使用 STORAGE=AUTO 选项。否则屏幕映像集中的多个符号映像会使用相同存储区域，造成冲突。

SISO：　指明本屏幕映像集中可以使用双字节字符集（DBCS，包括汉字在内的多种文字的字符集）。映像集中的所有域都可以输出 DBCS，但只有表明 SISO 的域才能输入 DBCS。

3-2-4　DFHMDI 宏



常用选项：

SIZE：　指明屏幕映像的大小，其格式为 SIZE=(行数，列数)，SIZE 给出本映像涵盖的屏幕大小。

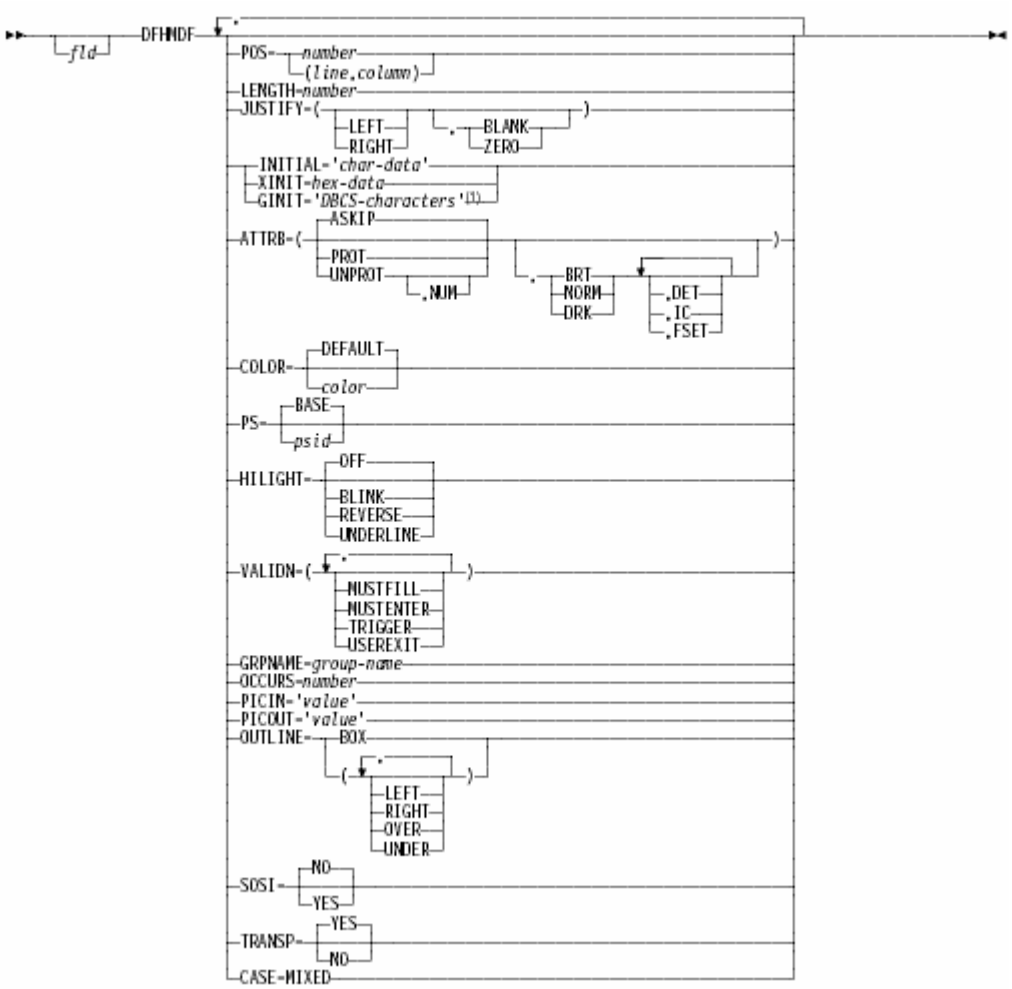
LINE：　表明本影响起始于屏幕的第几行，格式为 LINE=X。行号从 1 开始。

JUSTIFY：　指明映像在屏幕上是左对齐还是右对齐。

COLUMN：　指明这个屏幕映像放在那一列，也就是指明映像的左边界或右边界。若 JUSTIFY 参数指定的是右对齐，则列号从右向左数。

CTRL：　定义 3270 终端参数，DFHMDI 中的定义覆盖 DFHMSD 的定义。可用的选项有 PRINT 允许启动与终端相连的 3270 打印机；FREEKB 每次 SEND MAP 后解锁键盘允许输入。

3-2-5　DFHMDF 宏



常用选项：

POS：　表明域的起始位置，其格式为 POS=(x , y)，x 和 y 分别是起始行数和列数。每个域的第一个位置放的是此域的控制字节，是不可见的，第一个可见字符出现在紧靠控制字节的下一个位置 。

LENGTH：表明域中数据字节的长度，格式为 LENGTH=x，长度值里不包括属性字节长度。

INITIAL：给定域内容的初始值。格式为 INITIAL=“字符串”。

ATTRIB：　给定域的属性。域的属性可以有一个或多个，有如下选择：

ASKIP：　光标自动跳过此域，因此是不能输入的域

BRT：　指明此域高亮显示

DRK：　指明此域是否可显示，密码字段经常使用此属性。

FSET：　标明此域的值在下次读屏时能够输入进来。

IC：　光标自动插在此域里

NORM：　指明此域是正常属性

NUM：　指明此域必须输入数字

PORT：　指明此域受保护，不能修改

UNPORT：指明此域可以修改。

HILIGHT：　给定域的高亮显示属性，可选值有 OFF 不做高亮显示（默认值）；BLINK 闪烁显示；REVERSE 黑白反显；UNDERLINE 显示下划线

SOSI：　YES/NO 指明此域能否使用双字节字符集（DBCS）的值

VALIDN：　只对 8775 型终端有效，可选如下值：MUSTFILL，此域输入时必须填满；MUSTENTER，此域输入时不能空；TRIGGER，此域是触发域；USEREXIT，此域有 BMS 用户出口程序处理。

COLOR：　对支持彩色显示的终端，可以设置域的颜色。

PICIN：　只对 COBOL 和 PL/I 语言有效，指明数据输入的格式，格式描述是 COBOL 变量的描述格式，如 PICIN='9999V99'。

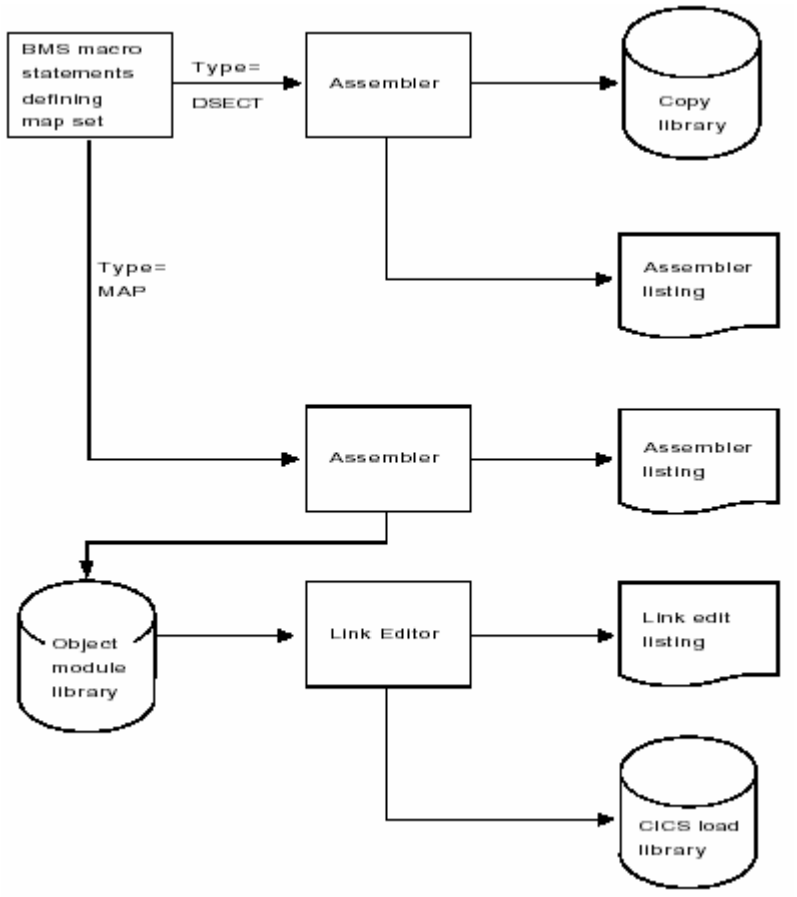
PICOUT：　类似 PICIN，给定数据输出时的格式

OUTLINE：　指定此域使用什么样的边框格式，可选值有：BOX，四周边框；LEFT，左边框；RIGHT，右边框；OVER，上边框；UNDER，下边框。

3-3　屏幕映像（MAP）的使用

- 字符映像和物理映像
- 屏幕映像的输出 SEND MAP
- 屏幕映像的输入 RECEIVE MAP

3-3-1 字符映像和物理映像



屏幕映像分为物理映像和字符映像，由汇编程序分别生成。

物理映像定义在 CICS PPT 表里面，文件放在 CICS 程序装载库里，当程序执行 SEND MAP 或 RECEIVE MAP 命令时，物理映像从库中加载执行。

字符映像由汇编器产生，使用 COBOL 语言的规范为使用屏幕映像定义的数据结构，应用程序通过它里面定义的变量，向屏幕映像输出或输入数据。物理映像与字符映像必须由同一个定义屏幕映像集的宏生成，才能保证一致性。

字符映像由汇编器根据 DFHMSD 宏定义的 MAPSET 生成，为每一个 MAP 生成一个输入格式的数据结构和一个输出格式的数据结构，为 MAP 里的每个 FILED，生成一个输入用的变量，一个输出用的变量，一个长度变量，一个标志变量和一个属性变量。

MAP 变量名由物理 MAP 名+后缀组成，后缀有两种“I”和“O”，“I”代表输入用的数据结构，“O”代表输出用的数据结构。

每个 MAP 的数据结构里，前 12 个字节是 TIOA(Terminal Input and Output Area)数据区。后面是各个域对应的变量，变量名称也是由域名+后缀组成，这些后缀的含义分别是：

输入 MAP 中：
L：存放域的长度，半字（2 字节）长。
F：存放域标志位。当操作者敲 EOF 键时，此位置 80，其他情况为 0。
I：存放用户在这个域中输入的数据。

输出 MAP 中：
A：存放域的属性字节，1 字节长。
O：存放输出到终端的数据

输入 MAP 和输出 MAP 共享同一段内存，对内存结构的定义不同。
下面就是前面给出的 MAPSET 定义对应的字符屏幕映像的 COBOL 代码：
USER01.SRC(TSTMPS1)

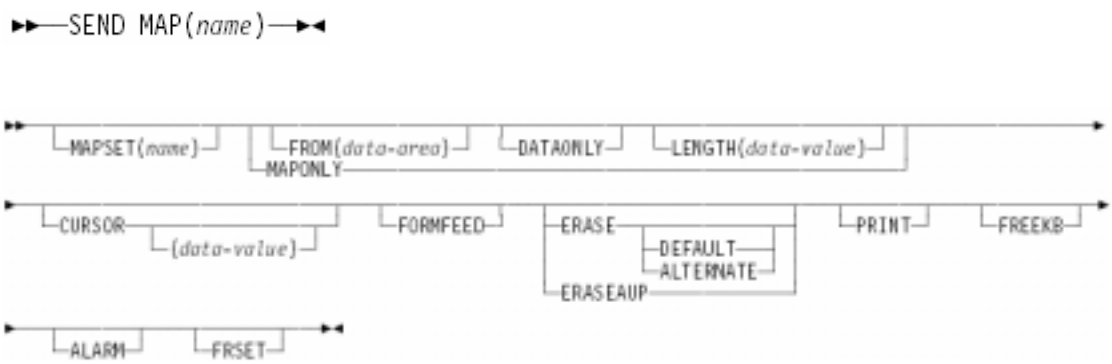
```
      01  TSTMP1I.
          02  FILLER PIC X(12).
          02  ACCTL    COMP  PIC  S9(4).
          02  ACCTF    PICTURE X.
          02  FILLER REDEFINES ACCTF.
            03 ACCTA    PICTURE X.
          02  ACCTI  PIC X(10).
          02  PWDL    COMP  PIC  S9(4).
          02  PWDF    PICTURE X.
          02  FILLER REDEFINES PWDF.
            03 PWDA    PICTURE X.
          02  PWDI  PIC X(6).
          02  MONEYL    COMP  PIC  S9(4).
          02  MONEYF    PICTURE X.
          02  FILLER REDEFINES MONEYF.
            03 MONEYA    PICTURE X.
          02  MONEYI  PIC X(6).
          02  REMARKL    COMP  PIC  S9(4).
          02  REMARKF    PICTURE X.
          02  FILLER REDEFINES REMARKF.
            03 REMARKA    PICTURE X.
          02  REMARKI  PIC X(20).
      01  TSTMP1O REDEFINES TSTMP1I.
          02  FILLER PIC X(12).
          02  FILLER PICTURE X(3).
          02  ACCTO  PIC X(10).
          02  FILLER PICTURE X(3).
          02  PWDO  PIC X(6).
          02  FILLER PICTURE X(3).
          02  MONEYO  PIC X(6).
          02  FILLER PICTURE X(3).
          02  REMARKO  PIC X(20).
```

3-3-1 字符映像和物理映像（续页 2）

```
       01  TSTMP2I.
           02  FILLER PIC X(12).
           02  OUTMSGL    COMP  PIC  S9(4).
           02  OUTMSGF    PICTURE X.
           02  FILLER REDEFINES OUTMSGF.
             03 OUTMSGA    PICTURE X.
           02  OUTMSGI  PIC X(60).
       01  TSTMP2O REDEFINES TSTMP2I.
           02  FILLER PIC X(12).
           02  FILLER PICTURE X(3).
           02  OUTMSGO  PIC X(60).
```

应用程序中一般只要把汇编生成的字符映像 COPY 到自己的 WORKING-STORAGE SECTION 里，通过这些变量就可以与终端上相应的域进行数据输入输出操作了。

3-3-2　屏幕映像的输出 SEND MAP



例：
```
EXEC CICS SEND MAP('TSTMP1')
     MAPSET('TSTMPS1')
     ERASE
     FREEKB
END-EXEC.
```

应用程序中把希望输出的数据内容填到字符屏幕映像的各个变量中去，就可以调用 CICS SEND MAP 命令把屏幕映像发送到终端显示。

SEND 命令选项：

MAP：　指定一个 MAP 名，这个 MAP 必须存在于下面指定的 MAPSET 中。

MAPSET(name)：　指定一个 MAPSET 名。这个 MAPSET 必须在 CICS 中定义并安装。

MAPONLY：　指明只发送物理 MAP 到终端，并不输出字符映像中的数据。

FROM(data-area)：　指定一个字符映像数据，里面包含希望送往屏幕的信息。

DATAONLY：　只输出字符映像中的数据，不输出物理映像。通常用于物理映像已经输出到屏幕上，用此选项更新一些域的内容。

LENGTH(data-value)：　半字数据，指明发往终端的数据长度。当希望发送数据长度比字符映像定义的长时，要指明此变量。

CURSOR(data-value)：　指明 MAP 显示结束后光标停放位置。

FORMFEED：　使 3270 打印机显示画面前先过页。

ERASE：　显示 MAP 前清屏。

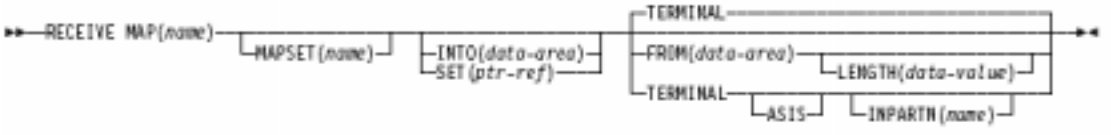
PRINT：　启动 3270 打印机

FREEKB：　完成 MAP 显示后解锁键盘。用户可以立即输入。

ALARM：　显示 MAP 时终端发警报声。

FRSET：　重置所有域的修改标志位。

3-3-3　屏幕映像的输入 RECEIVE MAP



例：　　EXEC CICS RECEIVE MAP('TSTMP1')
　　MAPSET('TSTMPS1')
　　END-EXEC.

应用程序通过发送 CICS 命令 RECEIVE MAP 来把用户更改后的 MAP 数据输入到字符影响变量中来，然后根据这些变量的值进行处理。

命令的选项：

MAP：　　指定一个 MAP 名，这个 MAP 必须存在于下面指定的 MAPSET 中。
MAPSET(name)：　　指定一个 MAPSET 名。这个 MAPSET 必须在 CICS 中定义并安装。
INTO(data-area)：　　指明输入数据存放变量，默认时放入 MAP 名+后缀“I”的变量里。
SET(ptr-ref)：　　把输入数据的地址放入给出的变量里。
TERMINAL：　　指明从与交易相连的终端读取输入。
FROM(data-area)：　　给出一个数据区，本命令从此数据区提取数据。

3-4　使用 BMS 程序的编译与执行

- 屏幕映像编译
- 应用程序编译
- 程序需要的配置与运行

3-4-1　屏幕映像编译

USER01.PROC(MAPPROC)　　MAP 编译 JCL 过程

```
//MAPPRO  PROC LOADLIB='EDU.ONLINE.LOADLIB',    生成物理映像存放的数据集(同执行码存放位置)
//             COPYLIB='USER01.COPY',           生成的字符映像文件存放库明
//             SRCLIB='USER01.BMSMACRO',        映像定义宏所在库
//             MACLIBC='CICSTS13.CICS.SDFHMAC',  CICS 库
//             MACLIBA='SYS1.MACLIB',
//             MAP=                             MAPSET 名
//COPY     EXEC PGM=IEBGENER                    把映像定义宏拷贝成临时文件
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=&SRCLIB(&MAP),DISP=SHR
//SYSUT2   DD DSN=&&TEMPM,UNIT=3390,DISP=(,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//            SPACE=(400,(50,50))
//SYSIN    DD DUMMY
//ASMBMS   EXEC PGM=ASMA90,PARM='SYSPARM(MAP),DECK,DBCS,NOOBJECT'  汇编 MAP
//SYSLIB   DD   DSN=&MACLIBC,DISP=SHR
//         DD   DSN=&MACLIBA,DISP=SHR
//SYSUT1   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSUT2   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSUT3   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSPUNCH DD DSN=&&BMS,DISP=(,PASS),UNIT=3390,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//            SPACE=(400,(50,50))
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DSN=&&TEMPM,DISP=(OLD,PASS)
//LKED     EXEC PGM=IEWL,PARM='LIST,XREF,LET',                    连接生成物理 MAP
//         COND=(5,LT,ASMBMS)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=&SYSUT1,SPACE=(1024,(120,120)),UNIT=3390
//SYSLMOD  DD DSN=&LOADLIB(&MAP),DISP=SHR
//SYSLIN   DD DSN=&&BMS,DISP=(OLD,DELETE)
//ASMCOP   EXEC PGM=ASMA90,PARM='SYSPARM(DSECT),DECK,NOOBJECT,DBCS', 汇编生成字符映像
//  COND=(5,LT,ASMBMS)
//SYSLIB   DD   DSN=&MACLIBC,DISP=SHR
//         DD   DSN=&MACLIBA,DISP=SHR
//SYSUT1   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSUT2   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSUT3   DD   UNIT=3390,SPACE=(CYL,(2,1))
//SYSPUNCH DD   DSN=&COPYLIB(&MAP),DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DSN=&&TEMPM,DISP=(OLD,DELETE)
```

编译 MAP 的 JCL

```
//MAPCOMP  JOB MSGLEVEL=(1,1),CLASS=A,REGION=2M,NOTIFY=&SYSUID
//PROCLIB  JCLLIB ORDER=USER01.PROC
//STEP1    EXEC MAPPROC,MAP='TSTMPS1',LOADLIB='EDU.ONLINE.LOADLIB',
//              COPYLIB='USER01.COPY',SRCLIB='USER01.SRC'
```

对编写的映像定义宏进行汇编，生成物理映像文件，放在 CICS 装载库里；生成字符映像文件，放到应用程序的 COPYLIB 里，供应用程序引用。

```
       IDENTIFICATION  DIVISION.
       PROGRAM-ID SAMPLE1.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  RESP-CODE    PIC S9(8)  COMP.
       01  RESP2-CODE   PIC S9(8)  COMP.
       01  MY-COMMAREA  PIC X(100).
       COPY TSTMPS1.
       LINKAGE SECTION.
       PROCEDURE DIVISION.
           IF EIBCALEN = 0 THEN
              MOVE LOW-VALUE TO TSTMP1O
              MOVE '0000000001' TO ACCTO
              EXEC CICS SEND MAP('TSTMP1')
                   MAPSET('TSTMPS1')
                   ERASE
                   FREEKB
              END-EXEC
              MOVE '1' TO MY-COMMAREA
              EXEC CICS RETURN TRANSID(EIBTRNID)
                        COMMAREA(MY-COMMAREA)
                        LENGTH(100)
              END-EXEC
           ELSE
              EXEC CICS RECEIVE MAP('TSTMP1')
                        MAPSET('TSTMPS1')
              END-EXEC
              STRING 'WITHDRAW \' MONEYI ' FROM ' ACCTI
                      ' PASSWORD:' PWDI ' IS DONE'
                   DELIMITED BY SIZE INTO OUTMSGO
              EXEC CICS SEND MAP('TSTMP2')
                   MAPSET('TSTMPS1')
                   ERASE
                   FREEKB
              END-EXEC
              EXEC CICS RETURN END-EXEC
           END-IF.
           GOBACK.
```

上面是一个例子程序的源码，为了程序简单，省去了所有的错误判断代码。应用程序的编译使用前一章所使用的 JCL 和 JCL 过程。编译连接生成可执行代码放到 CICS 的装载库中。

本程序使用了伪会话方式，在一次数据输出，用户修改输入处理的过程中，本程序被调用两次，作为两个交易，共同完成一次会话。关于本程序是否第一次被调用，程序中是用判断 EIBCALEN 是否为零实现的。交易第一次调用时，没人把 COMMAREA 传递给他，所以 EIB 信息中的 EIBCALEN 是 0，这时程序先把一个 MAP 发送到屏幕，然后为 COMMAREA 第一个字段置一，接下来就调用 CICS RETURN 命令返回，指定下个交易为它本身，且传递 100 个字节的 COMMAREA 过去，这时第一个交易结束。当用户在 MAP 上更改了数据，按键提交时，CICS 会再次调用本交易，这时会有 100 字节的 COMMAREA 由 CICS 传递过来，里面是上一个交易修改过的数据，前后连个交易见通过这种方式来传递一些应用数据。第二个交易处理数据后，发送另一个 MAP 显示成功信息。

- 定义程序
- 定义交易
- 定义 MAP
- 安装程序
- 安装交易
- 安装 MAP

MAP 和程序都编译连接成功后还要在 CICS 中定义和安装这些资源，使用如下命令：

```
CEDA DEF PROG(BMS1) GROUP(TESTGP)
CEDA DEF TRANS(TRN2) GROUP(TESTGP) PROGRAM(BMS1)
CEDA DEF MAPSET(TSTMPS1)   GROUP(TESTGP)
CEDA INS PROG(BMS1) GROUP(TESTGP)
CEDA INS TRANS(TRN2) GROUP(TESTGP)
CEDA INS MAPSET(TSTMPS1)   GROUP(TESTGP)
```

然后就可以在 CICS 终端上键入交易代码 TRN2 来执行交易。需要注意的是和程序一样，每次重新汇编后，物理映像也要使用 CEMT 进行更新。

练习 2

第四章　处理外部数据

- 处理 VSAM 文件数据

- 处理 DB2 关系数据库数据

4-1　处理 VSAM 文件数据

- VSAM 文件
- VSAM 文件记录查询 READ 命令
- VSAM 文件记录浏览
- VSAM 文件记录更新
- 添加 VSAM 文件记录
- 删除 VSAM 文件记录

VSAM 文件：

VSAM(Virtual Storage Access Method)文件是 IBM 在虚拟存储和树型数据结构基础上，为满足数据量大，存取速度快和维护方便而发展起来的一种文件组织形式。VSAM 以索引键（KEY）或相对字节地址（BRA）来安排记录的存放位置，它支持直接访问和顺序访问两种方式。

VSAM 文件结构：

VSAM 文件中的数据都以记录的格式存放，逻辑记录是访问数据的单位。VSAM 文件将记录存放在一个个 CI（Control Interval）中，一个 CI 是 DASD 中一片连续的区域，用来存储数据记录及控制信息。当读取一个记录时，这个记录所在的 CI 将整个读到 VSAM I/O 缓冲区，然后用户要读的记录才从 VSAM 缓冲区传输到用户定义工作区。CI 的大小在创建 VSAM 文件时由用户指定，也可以让系统自动选择合适的大小。

每个 CI 含有如下信息：

逻辑记录：记录用户数据，每个 CI 可以含有多个逻辑记录

自由空间：每个 CI 中可以有一定的自由空间，可以用来插入新的纪录。

控制信息：主要是 RDF 和 CIDF，RDF 描述每一个记录的信息，而 CIDF 则描述整个 CI 的信息。

CICS 支持一下三种 VSAM 文件：

- ✓ KSDS (Key Sequenced Data Set)
- ✓ ESDS（Entry Sequenced Data Set)
- ✓ RRDS (Relative Record Data Set)

VSAM 文件的定义：

VSAM 数据集可以通过访问方法服务（Access Method Services）的 DEFINE CULSTER 或 ALLOCATE 命令来定义。定义 VSAM 时要指定数据集的属性。如果系统中使用了 SMS 来管理存储，定义的数据集又是系统管理的数据集，则可指定 DATA CLASS，MANAGEMENT CLASS 和 STORAGE CLASS，也可以使用系统默认的存储定义。

VSAM 数据集的属性信息有：

INDEXED|NONINDEXED|NUMBERED|LINEAR：指定数据集的形式分别是 KSDS|ESDS|RRDS|LDS

RECORDSIZE：指定记录的平均和最大长度。

KEY：指明 KSDS 中键值的长度

CATALOG：指明所用编目的名字和口令

VOLUMES：指明所用的卷

REDORDS|KILOBYTES|MEGABYTES|TRACKS|CYLINDERS：以不同方式指定分配空间的大小。

BUFFERSPACE：	指明处理该数据集时应分配的最小缓冲区。
CONTROLINTERVALSIZE：	指明 CI 的大小。
SPANNED：	指明 CI 可否跨 CI 存放
IMBED：	指示是否把索引的顺序集放在相应的 CA 中。
FREESPACE：	指明预留自由空间大小。
SHAREOPTION：	指明文件在不同 REGION 间共享级别

一个定义 VSAM 的 JCL 例子：

```
//DEFVSAM  JOB NOTIFY=&SYSUID,MSGLEVEL=(1,1),MSGCLASS=X
//STEP1    EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE DEVP124.VSAMDATA  CLUSTER PURGE
 DEFINE CL(NAME(DEVP124.VSAMDATA)      -
          CYLINDERS(1   1  ) -
          SHAREOPTIONS (2) -
          RECORDSIZE (80    80   ) -
          INDEXED -
          KEYS( 10  0 )
/*
```

做好的 VSAM 文件要在 CICS 中使用，还必须在 CICS 中定义

例：

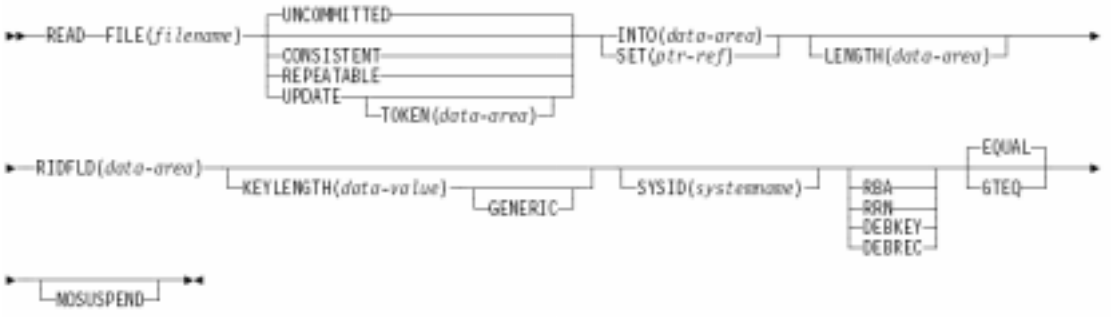
CEDA DEF FILE(SAVING) DSNAME(DEVP124.VSAMDATA) GROUP(TESTGP)

CEDA INS FILE(SAVING) GROUP(TESTGP)

一个文件不能同时在 CICS 中和 CICS 外同时打开，如果想用 DITTO 查看，必须先在 CICS 中把它关闭，如：CEMT SET FILE(SAVING) CLOSE

再次在 CICS 中使用时要把它打开：CEMT SET FILE(SAVING) OPEN

4-1-2　VSAM 文件记录查询 READ 命令



CICS 中使用 VSAM 文件要预先定义并打开，给 VSAM 文件指定一个名字，程序使用此文件名来引用。查询 VSAM 文件要使用记录标志域（RIDFLD），对不同类型 VSAM 文件 RIDFLD 有不同含义：

KSDS：RIDFLD 是文件的键值，可以是完整的键值，也可以是键值的一部分。

ESDS：RIDFLD 是一个 4 字节的相对字节数（RBA），指明记录离文件开始的偏移。

RRDS：RIDFLD 是一个 4 字节的相对记录号(RRN)

CICS 中使用 READ 命令来把符合要求的数据读入用户数据区中。

READ 命令选项：

FILE(filename)：　要查询的文件名字，文件要在 CICS FCT 里有定义。

UNCOMMITTED：　允许读未 COMMIT 的纪录。

CONSISTENT：　读一致性保证，在命令执行期间在记录上加 SHARE 锁。

REPEATABLE：　读一致性保证，在整个 UOW 期间在记录上加 SHARE 锁。

UPDATE：　通知 CICS 读取的纪录将用于修改或删除，用于保证数据一致性。

INTO(data-area)：　给出接收数据的工作区。

SET(ptr-ref)：　把指针 ptr-ref 指向读到的数据区。

LENGTH(data-area)：半字长二进制数据，存放读取纪录长度。读取完成后，里面是实际读出的记录长度。

RIDFLD(data-area)：　见上面描述。

KEYLENGTH(data-value)：半字长二进制变量，存放 KEY 长度。

SYSID(systemname)：给定 CICS REGION 名字，指明文件在那个 REGION 上。

RBA：　指明前面给出的 RIDFLD 是个 RBA

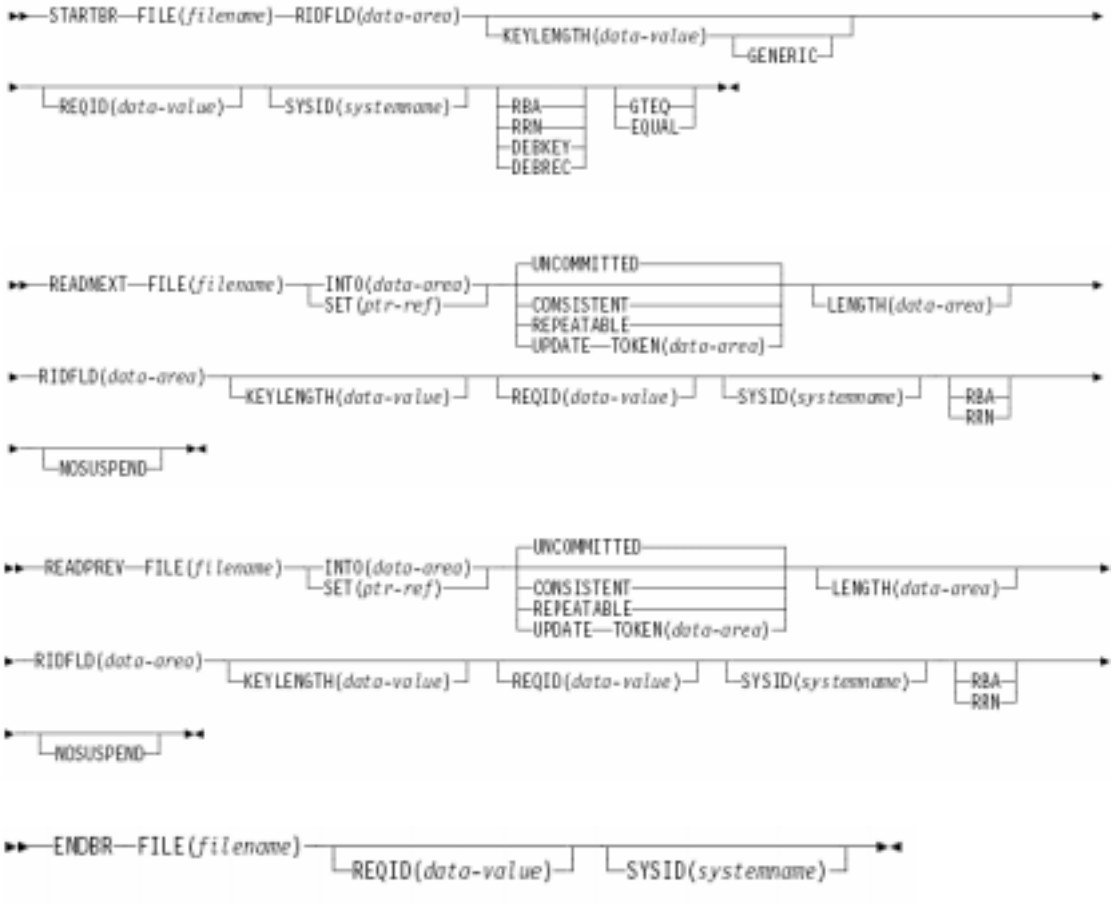
RRN：　指明前面给出的 RIDFLD 是个 RRN

NOSUSPEND：　如果要查询的纪录正被锁定，不等待，立即返回。

异常：

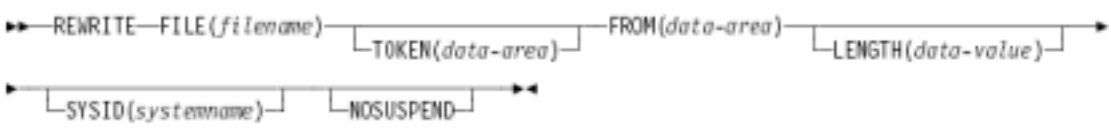
NOTFND：	指定键值的纪录没有找到
ENDFILE：	已到文件结尾，没有找到
INVREQ：	非法请求
FILENOTFOUND：	文件没有在 CICS 里定义，FCT 中找不到
ILLOGIC：	其他 VSAM 文件错误
IOERR：	文件 I/O 错误
DISABLED：	文件已被禁用
NOTOPEN：	文件已关闭

4-1-3　VSAM 文件记录浏览



对 VSAM 文件可以从头开始，依次查询各条记录，这要用到 4 条 CICS 命令：STARTBR、READNEXT、READPREV、ENDBR，依次为开始浏览，读下条记录、读前条记录和结束浏览命令。

这些命令使用的选项含义与 READ 命令大致相同，但增加了一项 REQID，这个选项用来使程序能对一个文件同时发动多个查询，各个查询可以有不同的当前位置。在 STARTBR 时，返回打开的浏览的 REQID，在后面的命令中使用 REQID 指明进行那个浏览的操作。程序通过检查命令的返回码来判断文件结束等情况。



CICS 使用 REWRITE 命令来更改数据，REWRITE 命令一定要跟在一个 READ UPDATE 命令之后，而且 KSDS 的 KEY 是不能改变的。REWRITE 命令的选项前面都介绍过了，它的异常情况也与 READ 大致相同，但多了以下几项：

LENGERR： 写入记录的长度没有指定或太长。

NOSPACE： 文件空间不足。可能的原因是文件中变长纪录长度增加了。

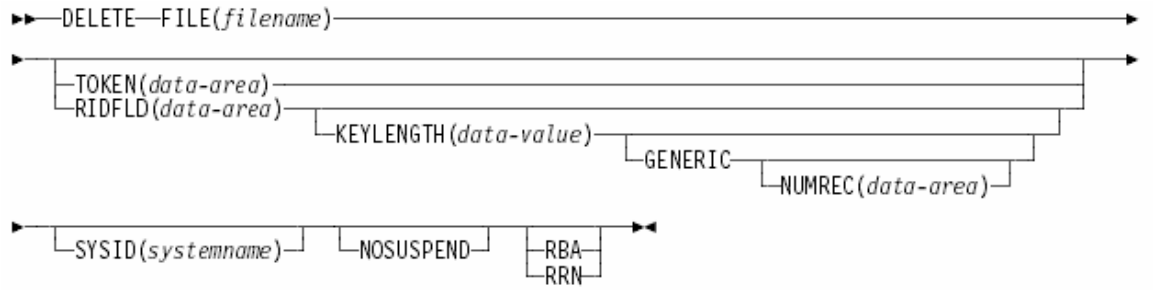
INVREQ： 多由以下原因造成：在 REWRITE 一个记录前，没有运行过 READ UPDATE 命令

ILLOGIC： 其他 VSAM 文件操作错误。这种情况下可以通过检测 EIBRCODE 字段来判断错误种类：在 EIBRCODE 中，第二个字节为 VSAM 返回码，第三个字节为 VSAM 错误码。具体代码可以到 VSAM 代码手册中查询。

4-1-5　添加 VSAM 文件记录

CICS 使用 WRITE 命令来向 VSAM 文件写入记录。命令选项含义与前面的介绍相同。当我们需要按键值的升序排列插入多个记录时，须使用 MASSINSERT 选项，每个 WRITE 命令都要包含这个选项，要终止 MASSINSERT 时，可通过 UNLOCK 命令来实现。

4-1-6　删除 VSAM 文件记录



CICS 使用 DELETE 命令来删除 VSAM 文件的记录。只有 KSDS 和 RRDS 结构的 VSAM 文件中的记录能被删除，这与数据的存放格式有关。ESDS 文件的记录按进入的顺序排放，只能在文件尾增加记录，不能删除，而且更新时不能改变长度。如果该纪录在请求删除前正在被更新，那么参数 RIDFLD 可以不指定，默认为当前记录。如果使用 GENERIC 参数，必须给定 KELENGTH 参数，指出 KEY 长度，同时 NUMREC 指出要删除的记录条数，这样就可以一次删除若干条记录。

练习 3

4-2　处理 DB2 关系数据库数据

- DB2 关系数据库
- 数据库数据查询
- 数据库数据更新
- 添加数据库数据记录
- 删除数据库数据记录

4-2-1　DB2 关系数据库

很多 CICS 系统应用中，数据是使用关系数据库来保存的，系统提供一个连接 CICS 应用程序和数据库的接口，称为 CICS Attachement Facility，程序通过它，使用 SQL 语句向 RDBMS 发出数据请求，这就为程序提供了更加方便灵活的数据控制手段。

程序使用 EXEC SQL 的方式把 SQL 语句嵌入进来，在程序的编译过程中在翻译 CICS 命令之前，要加上 DB2 的预编译，把 EXEC SQL 命令翻译成宿主语言的语句；还要增加 DB2 的 BIND 操作。

同批量的 DB2 程序一样 CICS 程序中要：

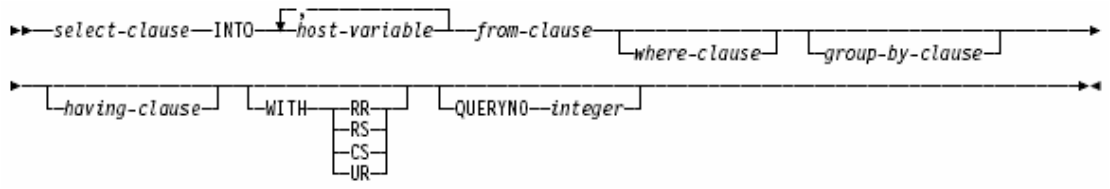
程序使用的表格和视图必须先在 WORKING-STORAGE SECTION 中声明

程序使用 SQLCA 来取得 SQL 语句执行结果情况

宿主变量必须在 EXEC SQL DECLARE 来定义

COBOL 中 SQL 语句以 EXEC SQL 。。。。 END-EXEC.格式嵌入。

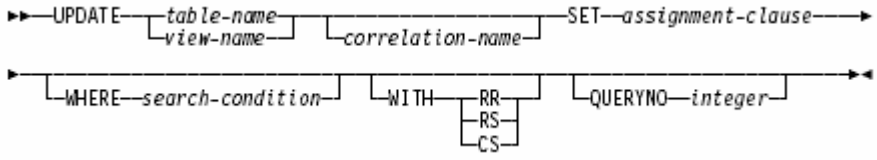
CICS 中的 DB2 程序不要使用 EXEC SQL COMMIT 和 EXEC SQL ROLLBACK 语句，而要代以 CICS 的数据操作语句 EXEC CICS SYNCPOINT 和 EXEC CICS ROLLBACK。



例：

```
EXEC SQL SELECT  NAME,PASSWD,BALENCE
             INTO  :WK-NAME,:WK-PASSWD,:WK-BAL
             FROM SAVING
             WHERE ACCOUNT=:WK-ACCT
END-EXEC.
IF SQLCODE NOT = 0
    ...
```

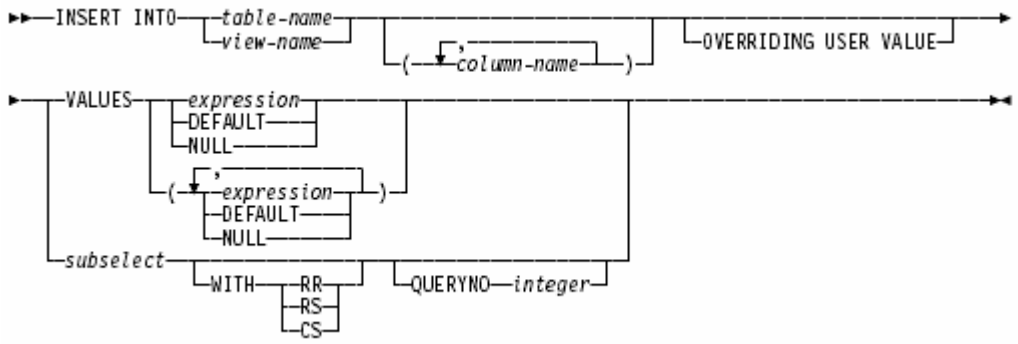
CICS 程序可以使用 SELECT 语句来查询 DB2 数据，SELECT 语法和具体编成用法请参考 DB2 程序设计手册。程序可以直接使用 EXEC SQL SELECT 。。。命令来查找单个的数据记录，也可以用 CURSOR 的手段来进行符合条件的若干记录的查找。



例：

```
EXEC SQL UPDATE SAVING
            SET BALENCE=:WK-BAL
            WHERE ACCOUNT=:WK-ACCT
END-EXEC.
IF SQLCODE NOT = 0
    ...
```

程序使用 UPDATE 语句去更新数据，UPDATE 语句详细语法参见 DB2 SQL 参考。

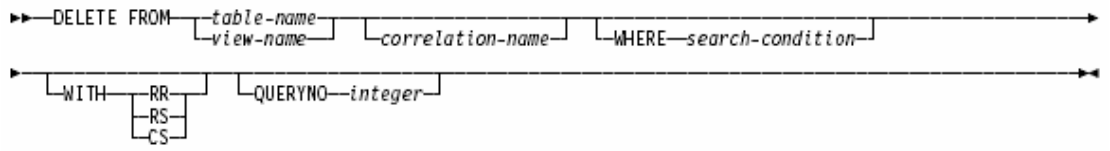


例：

```
EXEC  SQL  INSERT  INTO  SAVING
           (ACCOUNT,NAME,PASSWD,BALENCE)
          VALUSE(:WK-ACCT,:WK-NAME,:WK-PASSWD,:WK-BAL)
END-EXEC.
IF SQLCODE NOT = 0
    ....
```

程序使用 INSERT 语句去添加数据，INSERT 语句详细语法参见 DB2 SQL 参考。

4-2-5　删除数据库数据记录



例：

```
EXEC SQL DELETE FROM SAVING
    WHERE ACCOUNT=:WK-ACCT
END-EXEC.
IF SQLCODE NOT = 0
```

程序使用 DELETE 语句去删除数据，DELETE 语句详细语法参见 DB2 SQL 参考。

第五章　程序与内存管理

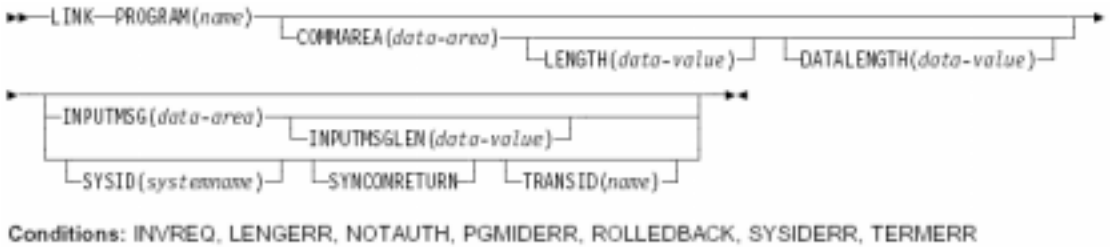
- 程序管理
- 交易管理
- 内存管理

5-1　程序管理

完成一个交易所需要的数据处理代码可以不必完全写在一个程序里，根据其逻辑内容分解成若干部分，每一部分写在一个程序里面，在交易里根据需要调用各个程序，这样做既可减少程序复杂度，又有利于程序复用。CICS 中调用程序有两种方式：LINK 和 XCTL

- LINK 命令
- XCTL 命令
- 使用 COMMAREA 进行数据传输
- 使用 INPUTMSG 进行数据传输
- LINK 命令与 XCTL 的差异

5-1-1　LINK 命令



例：

```
IDENTIFICATION DIVISION.
PROGRAM ID. PROG1.
...
WORKING-STORAGE SECTION.
01 COM-REGION.
   02 FIELD PICTURE X(3).
...
PROCEDURE DIVISION.
    MOVE 'ABC' TO FIELD.
    EXEC CICS LINK PROGRAM(©PROG2©)
              COMMAREA(COM-REGION)
              LENGTH(3)
    END-EXEC.
    ...
```

使用 LINK 命令，在某个逻辑层次上运行的程序，可以把控制交给在下一个逻辑层次上运行的程序，当被连接程序处理完成，执行 RETURN 命令时，控制再次返回到调用程序手中。

命令选项：

PROGRAM(name)：　给出被调用程序名，此程序要在 CICS 中有定义而且已经安装。

COMMAREA(data-area)：　指定传递给被调用程序的数据区，被调用程序使用 DFHCOMMAREA 变量来引用此数据区。

LENGTH(data-value)：　半字长二进制参数，指定 COMMAREA 的长度。如果 COMMAREA 要在两个 CICS SERVER 间传递，它的长度不能超过 32500 字节。

DATALENGTH(data-value)：　半字长二进制参数，指定 COMMAREA 里从头起的一定的内存长度。调用程序时如果只传给被条用程序较短的数据，DATALENGTCH 可以给定一个较小的值，这样可以传给被调用程序较少数据，而当被调用程序返回时，可以返回更多的数据。本参数主要为提高系统效率而设置。

INPUTMSG(data-area)：　指定一个传递给被调用程序的数据区，被调用程序可以使用 RECEIV 命令得到它。

INPUTMSGLEN(data-value)：　半字长二进制参数，指定 INPUTMSG 的长度。

SYSID：　指明被调用程序所在的 CICS REGIONS 名

SYNCONRETURN：　在指定 SYSID 时，当程序在指明的 REGION 正常完成，返回前下一个 SYNCPOINT。

TRANSID(name)：　指明当调用远程程序时，在远程使用的镜像交易名称，如果没有指定，默认的是 CPMI

异常：

INVREQ：请求非法，根据 RESP2：

8　LINK 命令使用了 INPUTMSG 选项，但它没有相联系的终端

14 使用了 SYNCONRETURN 选项，但程序本身就是在镜像交易中运行。

15 程序已经在镜像交易中运行,但它的 TRANSID 选项又指定了不同的镜像交易名

16 使用了 TRANSID 但全是空格

30 程序控制器没有初始化

LENGERR：长度错误，根据 RESP2 :

11 COMMAREA 给定的长度小于 0 或大于 32767.

12 DATALENGTH 给的是负值.

13 DATALENGTH 给的长度比 LENGTH 给的还大.

26 COMMAREA 地址是 0,但给的长度却不是 0

27 INPUTMSG 长度小于 0 或大于 32767

NOTAUTH：权限不足

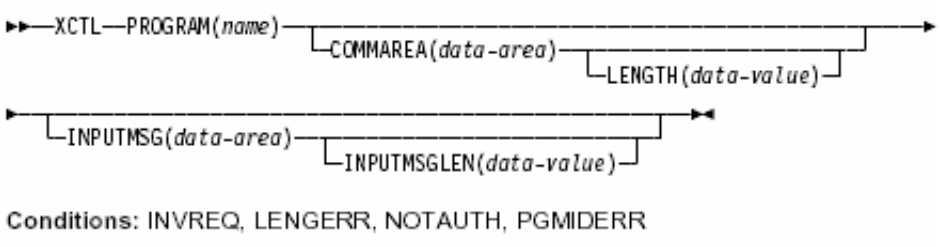
PGMIDERR：程序名错

ROLLEDBACK：指定了 SYNCONRETURN 选项，但远程交易失败

SYSIDERR：远程 REGION 名字错

TERMERR：终端错误。

5-1-2　XCTL 命令



例：

```
PROGRAM-ID. PROGA
WORKNG-STORAGE SECTION.
01WK-COMMAREA.
     05 WK-DATA    PIC X(8).
     05 WK-STATUS  PIC X VALUE  ‘N’.
     05 WK-DATA2   PIC X(31).
。。。
PROCEDURE DIVISION.
...
EXEC CICS XCTL PROGRAM( ‘PROGB’ )
          COMMAREA(WK-COMMAREA)
          LENGTH(40)
END-EXEC.
。。。
```

程序用 XCLT 命令把控制传给同一逻辑层次上的另一个程序，然后调用者程序退出。

命令选项：

PROGRAM(name)：　给出被调用程序名，此程序要在 CICS 中有定义而且已经安装。

COMMAREA(data-area)：　指定传递给被调用程序的数据区，被调用程序使用 DFHCOMMAREA 变量来引用此数据区。

LENGTH(data-value)：　半字长二进制参数，指定 COMMAREA 的长度。如果 COMMAREA 要在两个 CICS SERVER 间传递，它的长度不能超过 32500 字节。

INPUTMSG(data-area)：　指定一个传递给被调用程序的数据区，被调用程序可以使用 RECEIV 命令得到它。

INPUTMSGLEN(data-value)：　半字长二进制参数，指定 INPUTMSG 的长度。

异常：

INVREQ：	请求非法
LENGERR：	长度错误
NOTAUTH：	权限不足
PGMIDERR：	程序名错

调用程序：

```
IDENTIFICATION DIVISION.
PROGRAM ID.  ‘PROG1’.
...
WORKING-STORAGE SECTION.
01 COM-REGION.
   02 FIELD PICTURE X(3).
...
PROCEDURE DIVISION.
    MOVE ’ABC’ TO FIELD.
    EXEC CICS LINK PROGRAM( ‘PROG2’ )
         COMMAREA(COM-REGION)
         LENGTH(3)
    END-EXEC.
...
```

被调用程序：

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ’PROG2’.
...
LINKAGE SECTION.
01 DFHCOMMAREA.
   02 FIELD PICTURE X(3).
...
PROCEDURE DIVISION.
    IF EIBCALEN GREATER ZERO
    THEN
        IF FIELD EQUALS ’ABC’
         ...
```

CICS 程序可以在 LINK、XCTL 和 RETURN TRANSID 命令中使用 COMMAREA 在程序间传递数据。调用程序的 COMMAREA 可以在 WORKING-STORAGE SECTION 里定义，也可以是 LINKAGE SECTION 里的变量。被调用程序中必须在 LINKAGE SECTION 里定义“DFHCOMMAREA”变量名来接受数据。COMMAREA 的数据内容和格式由两个程序自己定义，接受程序定义 COMMAREA 的长度可以和发送程序不同，如果它只需要较少的数据 ，可以定义较短的变量，但是不能比发送方的更长。COMMAREA 区的实际长度可以通过查询 EIB 变量 EIBCALEN 得到，如果没有 COMMAREA，EIBCALEN 等于 0。对于 LINK 命令当被调用程序发出 RETURN 命令时，控制回到调用程序，修改过的 COMMAREA 也在调用程序的变量里有反映。

调用程序：

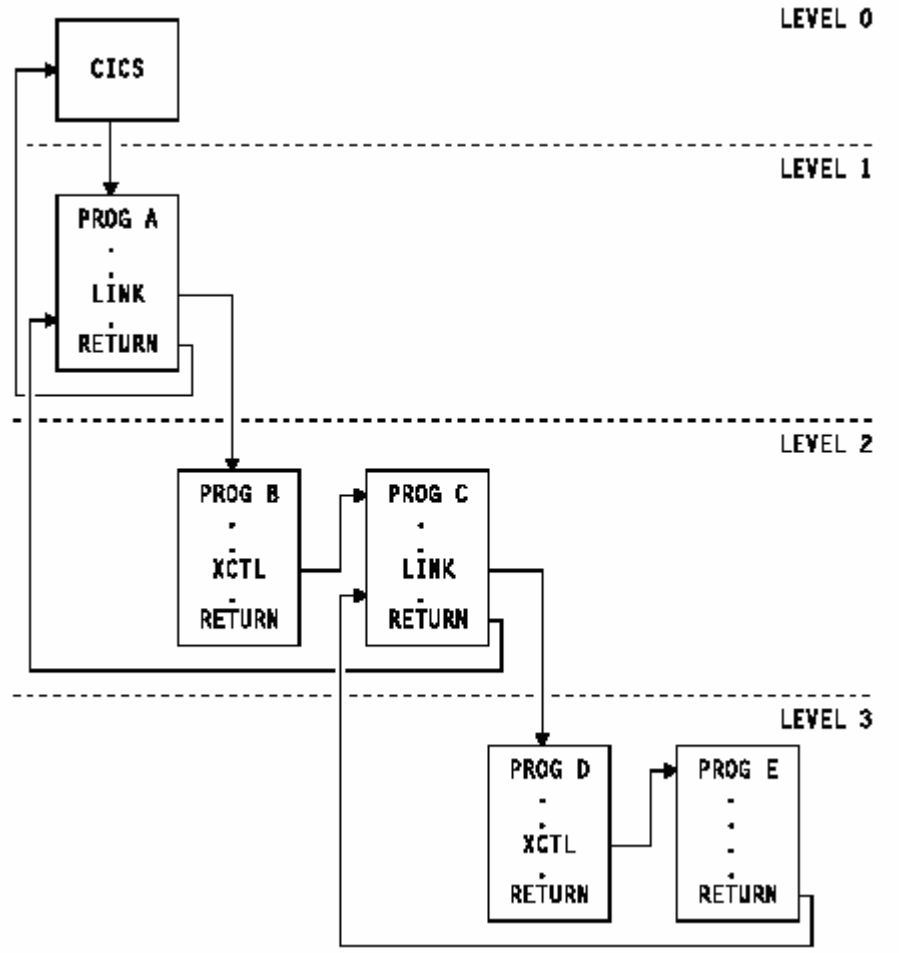
```
IDENTIFICATION DIVISION.
PROGRAM ID.  ‘PROG1’.
...
WORKING-STORAGE SECTION.
01 COM-REGION.
   02 FIELD PICTURE X(3).
...
PROCEDURE DIVISION.
    MOVE ’ABC’ TO FIELD.
    EXEC CICS XCTL PROGRAM( ‘PROG2’ )
         INPUTMSG(COM-REGION)
         INPUTMSGLEN(3)
    END-EXEC.
...
```

被调用程序：

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ’PROG2’.
 ...
 WORKING-STORAGE SECTION.
 01 FIELD    PIC X(3).
 01 FIELDLEN PIC S9(4) COMP.
 ...
 PROCEDURE DIVISION.
     MOVE 3 TO FIELDLEN.
     EXEC      CICS      RECEIVE
 INTO(FIELD)
               LENGTH(FIELDLEN)
     END-EXEC.
          ...
```

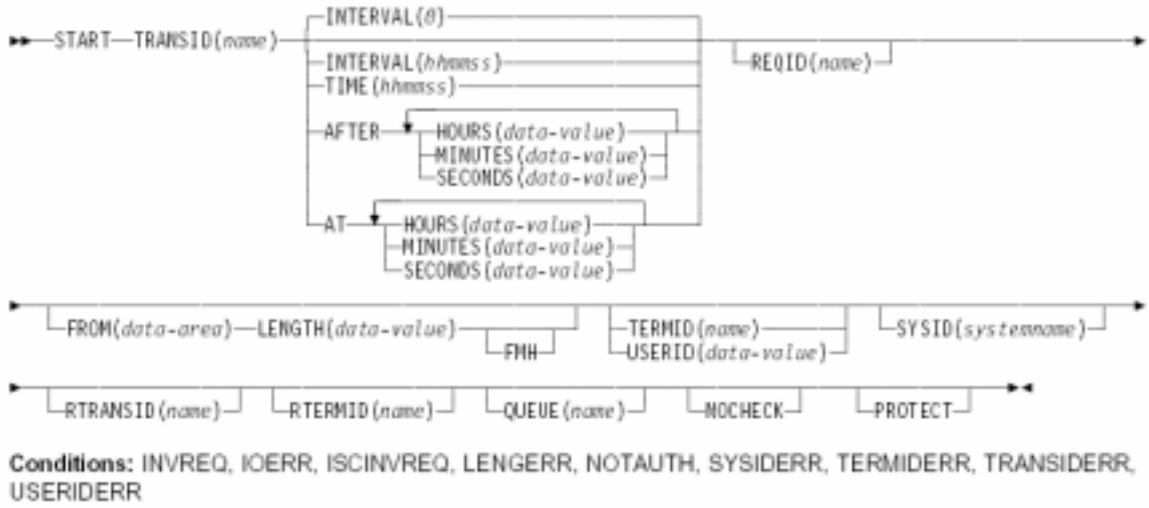
使用 INPUTMSG 是 LINK、XCTL 和 RETURN TRANSID 命令传递数据的另一个方法，被调用程序使用 RECEIVE 命令来得到数据。这种方式使程序员可以写这样的程序：它既可直接从终端发起，又可以用其他程序调用，只用相同的代码，不必分别处理。如果不是用 INPUTMSG，一旦调用程序使用 RECEIVE 读取了终端输入，被调用程序再发出 RECEIVE 命令时会停在那里等待输入。

CICS 程序间还有其他一些传输数据的手段，如使用 TWA（Transaction Work Area）、TDQ、TSQ 等，详见后述。



LINK 的程序在一个新的逻辑层次上运行，调用程序等待被调用程序返回。

XCTL 的程序在同一个逻辑层次上运行，调用程序发出 XCTL 后结束。



START 命令可以在本地或远程系统，立即或在指定的时间启动一个交易。程序可以向被发起程序传递数据，可以指定它的终端号等信息。

命令选项：

TRANSID(name)：	给出发起的交易名称，交易名 1-4 个字符，必须在 CICS PCT 表中有定义。
INTERVAL(hhmmss)：	给出交易启动的时间间隔，被发起交易在指定的时间之后启动。
TIME(hhmmss)：	指定交易启动时间，被发起的交易在指定的时间启动。
AFTER：	给出交易启动的时间间隔，被发起交易在指定的时间之后启动。
AT：	指定交易启动时间，被发起的交易在指定的时间启动。
REQID(name)：	指定一个唯一的名字，在交易达到启动时间之前，其他程序可以根据这个名字使用 CANCEL 命令来取消交易运行。
FROM(data-area)：	指定传给被发起交易的一个数据区
LENGTH(data-value)：	指定这个数据区长度
FMH：	指明传递给被发起交易的数据区包含 function management header
TERMID(name)：	指明被启动交易相联系的终端 ID
USERID(data-value)：	指明被启动交易的用户 ID
SYSID(systemname)：	指定被发起交易所在 CICS REGION 名字
RTRANSID(name)：	指定一个 1-4 字节的名字，被发起交易可以得到。
RTERMID(name)：	指定一个 1-4 字节的名字，被发起交易可以得到。
QUEUE(name)：	指定一个 TSQ 名字，被发起交易可以从中读取数据。

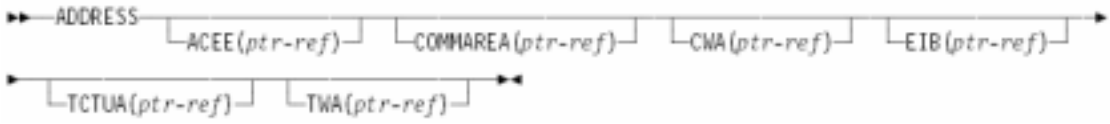
NOCHECK：　为提高效率 CICS 对本 START 命令作较少检查。
PROTECT：　被调用交易必须在调用者 SYNCPOINT 以后才能启动，若发起交易 ABEND，被发起交易取消。

异常：
INVREQ：　请求非法
IOERROR：　I/O 错误
LENGERR：　长度错误
ISCINVREQ：　远程系统发生不明错误
NOTAUTH：　权限不足
SYSIDERR：　远程 REGION 名字错
TERMIDERR：　终端 ID 错误。
TRANSIDERR：　交易码错误
USERIDERR：　用户代码错误

5-3　内存管理

- 地址定位命令 ADDRESS
- 获取主存空间 GETMAIN 命令
- 释放内存空间 FREEMAIN 命令

5-3-1　地址定位命令 ADDRESS



CICS 程序使用 ADDRESS 名令来得到程序可以访问的一些系统数据区的地址。

命令选项：

ACEE(ptr-ref)：　返回一个指向 ACEE（Access Control Environment Element）的指针，这个控制数据区是由外部安全管理器（ESM，External Security Manager）根据用户权限生成的。

COMMAREA(ptr-ref)：返回一个指向 COMMAREA 的指针。

CWA(ptr-ref)：　返回一个指向 CWA 的指针。CWA 是 common work area 的缩写它是一块在整个 CICS REGION 里都能访问到的数据区。其大小由 CICS 系统启动参数定义。

EIB(ptr-ref)：　返回 EIB 指针。

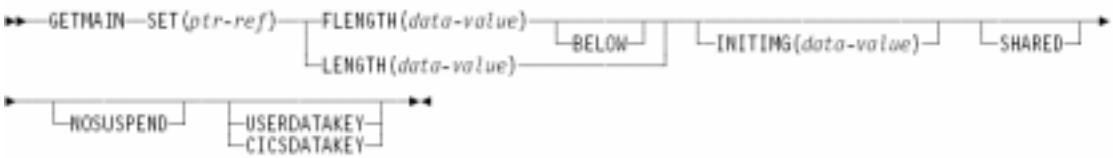
TCTUA(ptr-ref)：　返回一个指向 TCTUA (terminal control table user area)的指针，它可以在与同一个终端相联系的程序间传递数据。

TWA(ptr-ref)：　返回一个指向 TWA 的指针。TWA 是 transaction work area 的简称，TWA 是一块在同一个交易中的所有程序都能够访问的数据区，经常用来在同一个交易的程序间传递数据。它的大小由 CICS 的交易定义指定。

在 COBOL 语言中，要把获得指针的变量定义在 LINKAGE SECTION.里并且使用“ADDRESS”记号，如下例：

```
WORKING-STORAGE SECTION.
...
LINKAGE SECTION.
01 WK-TWA.
   05 TWA-FILED1  PIC X(5).
   05 TWA-FILED1  PIC X(15).
...
PROCEDURE DIVISION.
    EXEC CICS ADDRESS
              TWA(ADDRESS OF WK-TWA)
    END-EXEC.
```

5-3-2　获取主存空间 GETMAIN 命令



CICS 程序用 GETMAIN 命令来动态申请工作内存。

命令选项：

SET(ptr-ref)：　　给出接收申请成功内存的指针变量。

LENGTH(data-value)：　半字长二进制参数，给出申请内存的字节数。

FLENGTH(data-value)：全字长二进制参数，给出申请内存的字节数。

BELOW：　　指明申请 16M 线以下的内存。

INITIMG(data-value)：一个字节的二进制参数，CICS 用这个字节的值来初始化申请到的变量。

SHARED：　　指明要申请的是共享内存。对没有此选项申请到的内存，在交易退出时CICS 自动把它释放;增加此选项,则必须显式地用FREEMAIN 命令才能释放内存。共享内存是不同 TASK 间传递数据的一种手段。

NOSUSPEND：　　指明当 CICS 没有足够内存时,不进行等待,直接返回并触发 NOSTG 异常。

CICSDATAKEY：　　指明要从 CICS DSA 中申请 CICS-KEY 的内存。

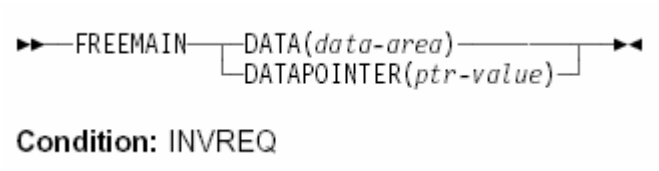
USERDATAKEY：　　指明申请的是 USER-KEY 的内存。

异常：

LENGERR：　长度错误

NOSTG：　没有足够内存

5-3-3　释放内存空间 FREEMAIN 命令



例：

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77 AREA-POINTER USAGE IS POINTER.
LINKAGE SECTION.
  01 WORKAREA PIC X(100).
PROCEDURE DIVISION.
    EXEC CICS GETMAIN SET(AREA-POINTER)
              LENGTH(100)
    END-EXEC.
    。。。
    SET ADDRESS OF WORKAREA TO AREA-POINTER.
    。。。
    EXEC CICS FREEMAIN DATA(WORKAREA) END-EXEC.
    EXEC CICS RETURN END-EXEC.
```

FREEMAIN 命令用来释放前面由 GETMAIN 命令得到的内存。一般的程序中使用 GETMAIN 获得的内存在 TASK 结束时会被自动释放，除非：

1) GETMAIN 命令使用了 SHARED 选项
2) 程序的定义里指定 RELOAD=YES
3) 程序定义指定 RELOAD=NO，但是使用 HOLD 选项被装载进来

这时就要程序发送 FREEMAIN 命令来释放申请到的内存。

命令选项：

DATA(data-area)：　　指明要释放的数据区。
DATAPOINTER(ptr-value)：　　指明要释放数据区的指针。

异常：

INVREQ: 根据 RESP2 的值:

1 给的内存不是使用 GETMAIN 命令得到的内存。
2 内存是用 CICS-key 参数申请的，程序却用 user-key 去释放它

练习 4

第六章　使用 CICS 队列

- TD QUEUE

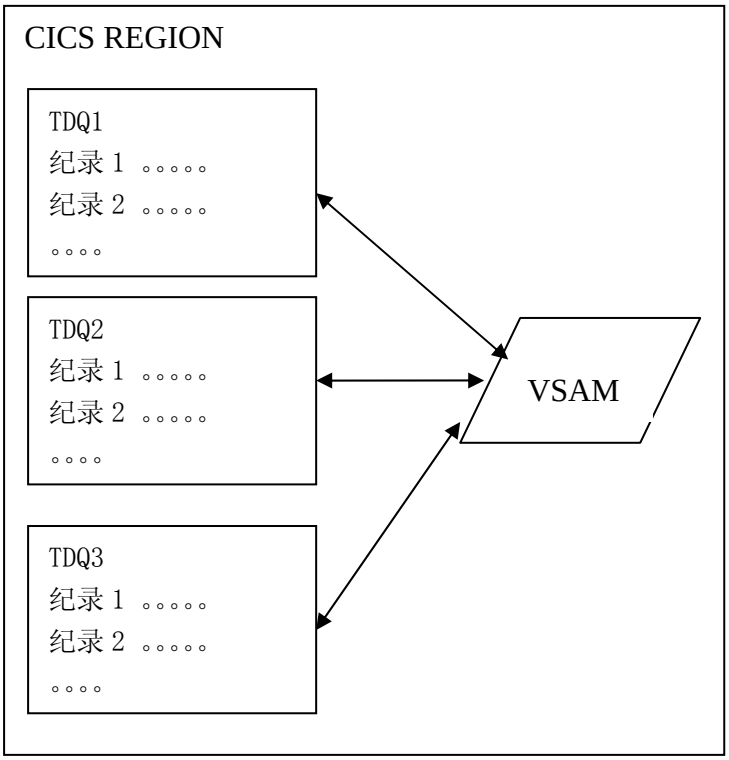
- TS QUEUE

6-1 TD QUEUE

CICS 的 TD（Transient Data）QUEUE 为应用程序提供了一种先入先出的数据结构。TD QUEUE 必须在 CICS 中提前定义，TDQ 的数据存储在数据文件里，是一种可恢复的 CICS 资源。

- 内部 TD QUEUE
- 外部 TD QUEUE
- TD QUEUE 的读取
- TD QUEUE 的写入
- TD QUEUE 的删除

6-1-1　内部（Intrapartition）TD QUEUE

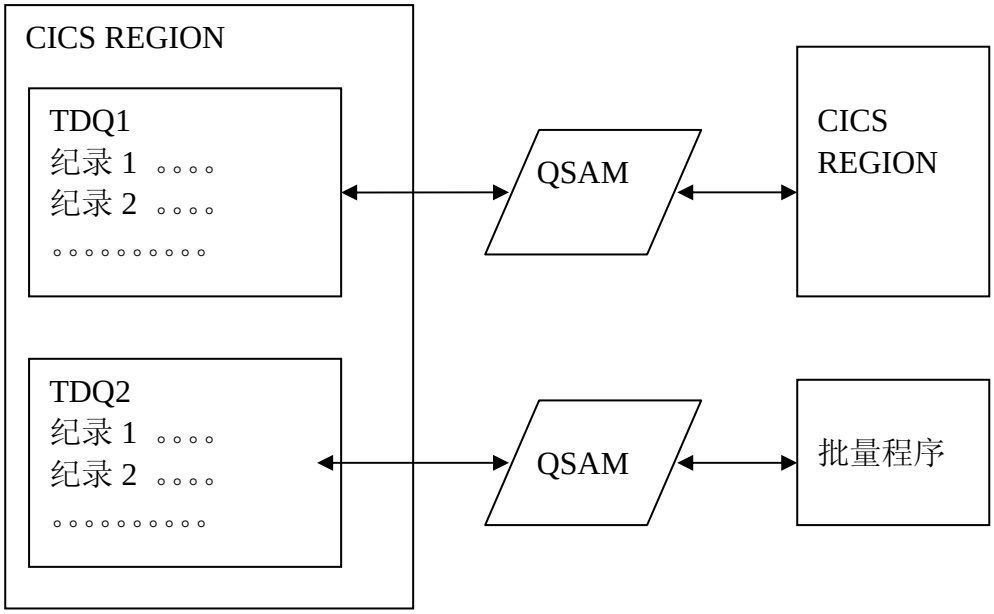


根据其资源定义，TDQ 有两种类型：内部（Intrapartition）TD QUEUE 和外部（Extrapartition）TD QUEUE。所有不同名字的内部 TDQ 的数据都存在同一个由 CICS 控制的 VSAM 文件里，只能在 CICS 内部使用；外部 TDQ 则是每个 TDQ 的数据都存在一个单独的顺序文件里，可以被批量程序读取处理。

内部 TDQ 在其资源定义时指定，每个 QUEUE 由一个最多 4 个字节的名字标识，每个 QUEUE 里可以存放若干条数据，每条数据可以具有不同的长度，记录按先入先出的规则被写入和读取，每条记录只能被读出一次。

内部 TD QUEUE 的特点；
所有 QUEUE 的数据都记录在一个 CICS 管理的 VSAM 文件里。
记录是可变长度的格式。
每个队列的纪录按次序写入。
队列里的纪录只能被读取一次，一旦记录被任一个程序读出，其他程序就不能够再读到它。
可触发自动任务初始化（Automatic Task Initial)
记录数据不可更改

6-1-2　外部（Extrapartition）TD QUEUE

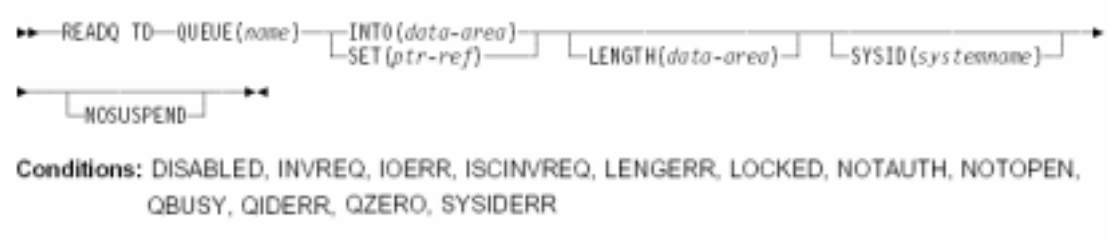


外部 TDQ 的数据存储在 CICS 之外的顺序数据集里，每个 TDQ 对应一个单独的 QSAM 文件，这个文件同时可以被其他 CICS REGION 或批量程序使用。TDQ 一般用来存储需要与 CICS 之外的程序共用的一些数据。应用程序不能删除外部 TDQ，否则将返回一个 INVREQ 的异常。

外部 TDQ 的特点：

1) 外部 TDQ 使用 QSAM 方式存储文件。
2) 每个外部 TDQ 使用一个单独的物理文件。
3) 每个外部 TDQ 都可以定义为用来输入或输出，但不能定义为同时输入和输出。
4) 每个外部 TDQ 是一个文件，所以可以单独控制一个 TDQ 的打开或关闭。
5) 外部 TDQ 里可以存放定长或变长纪录。
6) TDQ 的数据文件可以由批量程序访问。

6-1-3　TD QUEUE 的读取



READ TD QUEUE 命令用来把 TDQ 里的数据读到程序变量中来。

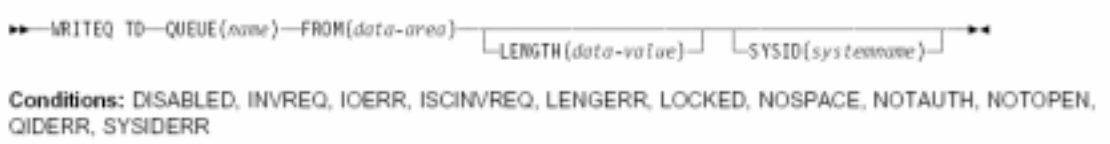
命令选项：

QUEUE(name)：	队列名称，1-4 个字符，队列必需在 CICS 的 DCT 中有定义。
INTO(data-area)：	指定接收的数据区。
SET(ptr-ref)：	指明接收数据的指针。
LENGTH(data-area)：	半字长二进制参数，指明接收数据区的长度。
SYSID(systemname)：	指明队列所在远程 CICS 系统的名字。
NOSUSPEND：	指明当被读 QUEUE 正被其他程序打开写入时，一直要等待直到可以读取，否则将返回 QBUSY 异常。

异常：

DISABLED：	队列被禁用
INVREQ：	请求非法
IOERR：	I/O 错误
ISCINVREQ：	远程系统上操作错误
LENGERR：	实际数据比给出长度长
LOCKED：	与队列有关的一个 UOW 陷于 in-doubt 状态。
NOTAUTH：	用户权限不足
NOTOPEN：	队列没有打开（只对外部队列）
QBUSY：	被读取的内部队列正在被其他程序打开写入或删除
QIDERR：	队列名错
QZERO：	队列空
SYSIDERR：	远程系统名错

6-1-4　TD QUEUE 的写入



WRITEQ TD 命令用来向 TDQ 里写入数据。

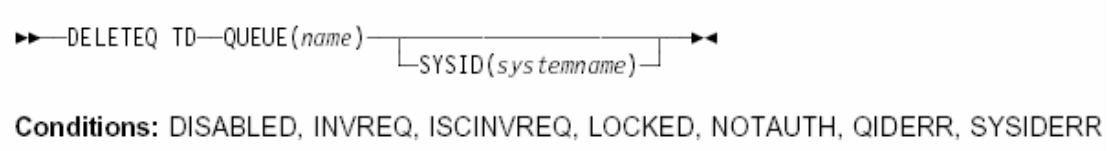
命令选项：

FROM(data-area)：　指定要写入 QUEUE 的数据区
LENGTH(data-value)：　半字长二进制参数，指定要写入的字节数
QUEUE(name)：　指定队列名，1-4 个字节，必须在 CICS 中有定义
SYSID(systemname)：　TDQ 所在远程系统名称

异常：

DISABLED：　队列被禁用
INVREQ：　请求非法
IOERR：　I/O 错误
ISCINVREQ：　远程系统上操作错误
LENGERR：　长度错误
LOCKED：　与队列有关的一个 UOW 陷于 in-doubt 状态。
NOTAUTH：　用户权限不足
NOTOPEN：　队列没有打开（只对外部队列）
NOSPACE：　空间满无法写入
QIDERR：　队列名错
SYSIDERR：　远程系统名错

6-1-5　TD QUEUE 的删除



DELETEQ TD 用来删除 TDQ 里的数据。外部 TDQ 的数据不能删除否则会引发 INVREQ 异常

命令选项：

QUEUE(name)：　　指定队列名，1-4 个字节，必须在 CICS 中有定义

SYSID(systemname)：　　TDQ 所在远程系统名称

异常：

DISABLED：　队列被禁用

INVREQ：　请求非法

ISCINVREQ：　远程系统上操作错误

LOCKED：　与队列有关的一个 UOW 陷于 in-doubt 状态。

NOTAUTH：　用户权限不足

QIDERR：　队列名错

SYSIDERR：　远程系统名错

6-2 TS QUEUE

- TS QUEUE
- TSQ 纪录的读取
- TSQ 纪录的写入
- TSQ 纪录清除

TSQ 是 Temporary Storage Queue 的缩写，它是放在主存或可直接访问的辅存内。每个 TSQ 有一个 1-8 个字节的名字，TSQ 不必预先定义，队列在第一个程序向其中写入数据时创建，每个 TSQ 中的纪录可以通过逻辑记录号来读取，直到队列被删除，TSQ 中的数据都不变，可以被读取任意次，即使写入 TSQ 的 TASK 结束了，其他 TASK 的程序仍然能够读取它写入的数据。TSQ 的数据是不可恢复的数据，当 CICS 再次启动时，原来在里面的数据将全部丢失。

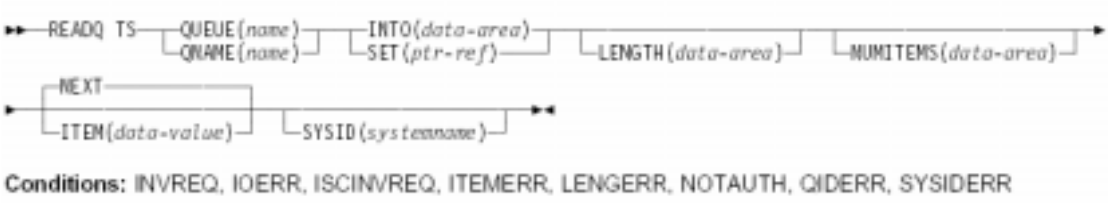
TSQ 的特点：

1) TSQ 中的纪录是变长的。
2) TSQ 不必预先在系统中定义，在第一个程序向其中写入数据时自动创建
3) 可以选择地要求一个可恢复队列，这时必须在 TST 中定义
4) TSQ 数据可以存储在内存或辅存中，这依赖于系统初始化时的定义和 WRITEQ TS 时的指定
5) 所有辅存的 TSQ 共享一个 VSAM ESDS 文件
6) 纪录可以顺序或直接读取，并且可以被更新
7) 读取操作不破坏纪录，直到显式地删除，数据一直可用
8) CICS 重启动将使 TSQ 中的数据丢失

TSQ 允许的操作：

WRITEQ TS：	写入 TSQ 纪录
WRITEQ TS REWRITE：	更新 TSQ 纪录
READQ TS：	从 TSQ 中读取一条纪录
READQ TS NEXT：	从 TSQ 中读取下条纪录
DELETEQ TS：	删除 TSQ 中的所有纪录

6-2-2　TSQ 纪录的读取



READQ TS 用来读取 TSQ 数据，TS 字样可用省略，READQ QUEUE(name)默认使用 TSQ。

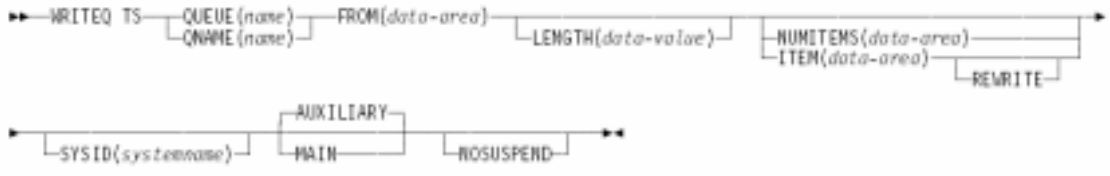
命令选项：

QUEUE(name)：	队列名称，1-8 个字符
QNAME(name)：	同 QUEUE，但可指定最长 16 个字节的名字
INTO(data-area)：	指定接收的数据区。
SET(ptr-ref)：	指明接收数据的指针。
LENGTH(data-area)：	半字长二进制参数，指明接收数据区的长度。
NUMITEMS(data-area)：	半字长二进制参数，用来返回 QUEUE 里一共有多少纪录。
ITEM(data-value) ：	半字长二进制参数，指明读第几条纪录。
NEXT：	指明读队列的下一条纪录（当前纪录可能是被任意一个 TASK 读取的）
SYSID(systemname)：	指明队列所在远程 CICS 系统的名字。

异常：

INVREQ：	请求非法
IOERR：	I/O 错误
ISCINVREQ：	远程系统上操作错误
LENGERR：	实际数据比给出长度长
ITEMERR：	无此纪录号
NOTAUTH：	用户权限不足
QIDERR：	队列名错
SYSIDERR：	远程系统名错

6-2-3　TSQ 纪录的写入



Conditions: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, LOCKED, NOSPACE, NOTAUTH, QIDERR, SYSIDERR

WRITEQ TS 用来向 TSQ 里写入纪录。

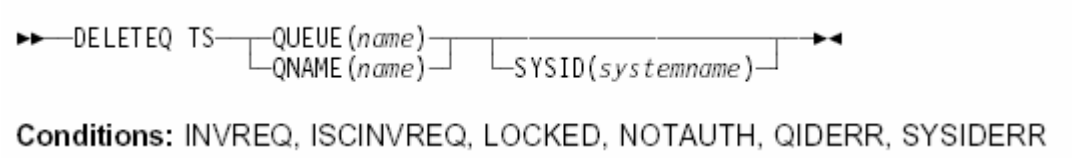
命令选项：

QUEUE(name)：	指定队列名，1-8 个字节
QNAME(name)：	指定 1-16 字节长的队列名字
FROM(data-area)：	指定要写入 QUEUE 的数据区
LENGTH(data-value)：	半字长二进制参数，指定要写入的字节数
ITEM(data-area)：	半字长二进制参数，指定要写入的纪录号（于 REWRITE 选项同时使用）
NUMITEMS(data-area)：	半字长二进制参数，CICS 用来返回写入后，队列里一共有多少纪录
REWRITE：	指明更新一条纪录的数据。
SYSID(systemname)：	TDQ 所在远程系统名称
MAIN：	指明 TSQ 数据存放在内存
AUXILIARY：	指明 TSQ 数据保留在辅存里
NOSUSPEND：	指明若 TSQ 数据空间不足，不进行等待，立即返回且引发 NOSPACE 异常

异常：

INVREQ：	请求非法
IOERR：	I/O 错误
ISCINVREQ：	远程系统上操作错误
LENGERR：	长度错误
LOCKED：	与队列有关的一个 UOW 陷于 in-doubt 状态。
NOTAUTH：	用户权限不足
NOSPACE：	空间满无法写入
QIDERR：	队列名错
SYSIDERR：	远程系统名错

6-2-4　TSQ 纪录清除



DELETEQ TS 用来清除 TSQ 的内容。

命令选项：

QUEUE(name)：	指定队列名，1-8 个字节
QNAME(name)：	指定 1-16 字节长的队列名字
SYSID(systemname)：	TDQ 所在远程系统名称

异常：

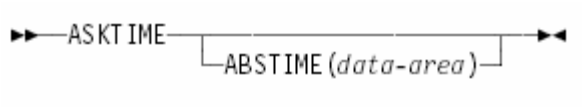
INVREQ：	请求非法
ISCINVREQ：	远程系统上操作错误
LOCKED：	与队列有关的一个 UOW 陷于 in-doubt 状态。
NOTAUTH：	用户权限不足
QIDERR：	队列名错
SYSIDERR：	远程系统名错

练习 5

第七章　其他常用的 CICS 命令

CICS 系统中有数量很大的各种 CICS 命令，为应用程序提供丰富的功能接口，详细情况请参照 CICS 应用程序参考手册，这里对一般程序中经常会用到的一些命令作一定的介绍。

- ASKTIME
- FORMATTIME
- ENQ
- DEQ
- DELAY
- CANCEL
- RETRIVE
- SYNCPOINT
- SYNCPOINT ROLLBACK



ASKTIME 命令返回当前系统时间。ASKTIME 命令更新 EIBDATE 和 EIBTIME 两个 EIB 变量，这两个变量的原来存放的是 TASK 启动时的时间。ASKTIME 接收到的时间以压缩十进制方式存放，不便使用，一般都再用 FORMATTIME 命令把时间转换成方便显示的形式。

命令参数：

ABSTIME(data-area)：指定接收时间的数据区，时间将以 packed decimal 方式存放，COBOL 中使用 PIC S9(15) COMP-3 形式的变量。

FORMATTIME 命令用来把绝对格式的时间转化为便于显示的形式。

命令选项：

ABSTIME(data-area)： 指定绝对时间，绝对时间是压缩十进制格式存放的系统时间，在 COBOL 中是 PIC S9(15) COMP-3 型变量

DATE(data-area)： 指定接收日期的变量，日期的格式据系统初始化时指定的 DATFORM 参数为准。

DATEFORM(data-area)： 指定日期格式，有"YYMMDD","DDMMYY"和"MMDDYY"选择。

DATESEP(data-value)： 指定在年月日间的分隔符。默认没有。

DAYCOUNT(data-area)： 全字长二进制参数，返回自 1900 年 1 月 1 日起的日数。

DAYOFMONTH(data-area)： 全字长二进制参数，返回本月第几日

DAYOFWEEK(data-area)： 全字长二进制参数，返回星期几，星期天=0，星期六=6

DDMMYY(data-area)： 8 字节字符变量，以 DD/MM/YY 方式返回日期，间隔符由 DATESEP 指定，若无间隔符，返回值左对齐。

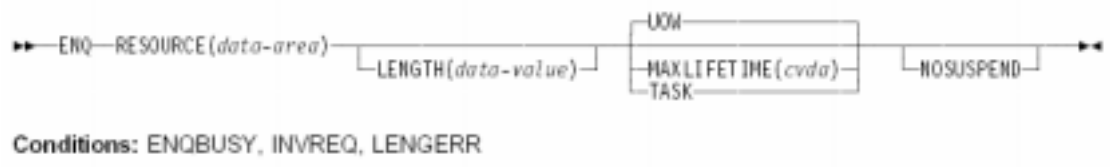
DDMMYYYY(data-area)： 10 字节字符变量，以 DD/MM/YYYY 方式返回日期，间隔符由 DATESEP 指定，若无间隔符，返回值左对齐。

FULLDATE(data-area)： 10 字节字符变量，以 DATFORM 系统初始化参数规定的格式返回日期

MMDDYY(data-area)： 8 字节字符变量，以 MM/DD/YY 方式返回日期，间隔符由 DATESEP 指定，若无间隔符，返回值左对齐。

MMDDYYYY(data-area)： 10 字节字符变量，以 MM/DD/YYYY 方式返回日期，间隔符由 DATESEP 指定，若无间隔符，返回值左对齐。

MONTHOFYEAR(data-area)：	全字长二进制参数，返回本年第几月。
TIME(data-area)：	8 字节字符变量，以"hh:mm:ss"格式返回时间，间隔符依 TIMESEP 选项指定。
TIMESEP(data-value)：	指定时间表示格式中时分秒间的间隔符。
YEAR(data-area)：	全字长二进制参数，返回年数。
YYDDD(data-area)：	6 字节字符变量，返回年和当年日数。间隔符由 DATESEP 规定。
YYDDMM(data-area)：	8 字节字符变量，返回 YY/DD/MM 格式日期,间隔符由 DATESEP 规定
YYMMDD(data-area)：	8 字节字符变量，返回 YY/MM/DD 格式日期,间隔符由 DATESEP 规定
YYYYDDD(data-area)：	8 字节字符变量，返回 YYYY/DDD 格式日期,间隔符由 DATESEP 规定
YYYYDDMM(data-area)：	10 字节字符变量，返回 YYYY/DD/MM 格式日期,间隔符由 DATESEP 规定
YYYYMMDD(data-area)：	10 字节字符变量，返回 YYYY/MM/DD 格式日期,间隔符由 DATESEP 规定



ENQ 语句用来在 TASK 间实现对某一资源使用的同步性。本命令里的“资源”是指命令提供的一个 1-255 字节的字符串，一旦一个 TASK 对某一资源发送了 ENQ 命令，其他 TASK 对同一资源的 ENQ 命令会失败或进入等待，直到此 TASK 对此资源调用 DEQ 命令释放为止。如果一个 TASK 没有发送 DEQ 命令，CICS 在那个 TASK 发 SYNCPOINT 或退出时会自动释放资源。

ENQ 和 DEQ 命令常用来实现不同 TASK 间对同一资源的同步控制，如对所有 TASK 公用的计数器加 1 的操作等。

命令选项：

RESOURCE(data-area)：指定要同步的资源，有两种方式给出资源，一种是指定一个数据区，它的地址就是这个资源；另一种是给出一个变量，它里面存放的数据就是资源，这时必须给定 LENGTH 选项。

LENGTH(data-value)：半字长二进制参数，制定资源数据的长度。

MAXLIFETIME(cvda)：指定 CICS 自动释放资源的条件，cvda 可以选择 UOW，资源在本 UOW 结束后自动释放；还可选择 TASK，资源在本 TASK 结束时自动释放。

TASK：MAXLIFETIME(TASK)的另一种写法。

UOW：MAXLIFETIME(UOW）的另一种写法。

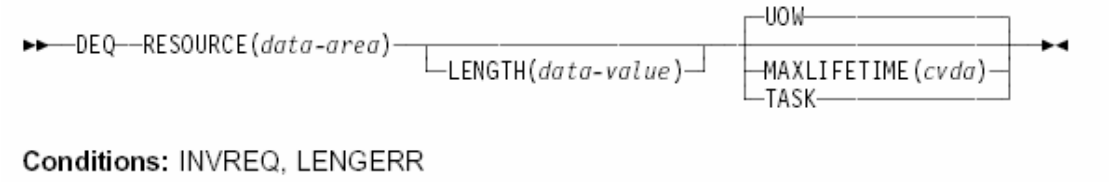
NOSUSPEND：指明当资源已被 ENQ 住时不进入等待，引发 ENQBUSY 异常。

异常：

ENQBUSY：资源已被其他 TASK ENQ 住。

INVREQ：非法请求。

LENERR：长度错误。



DEQ 命令用来释放被 ENQ 住的资源。DEQ 的资源必须已被 ENQ，否则命令会被忽略。若一个程序里对某资源使用了一次以上的 ENQ 命令，必须同等数目的 DEQ 命令才能把资源完全释放。

命令选项：

RESOURCE(data-area)：指定要同步的资源，有两种方式给出资源，一种是指定一个数据区，它的地址就是这个资源；另一种是给出一个变量，它里面存放的数据就是资源，这时必须给定 LENGTH 选项。

LENGTH(data-value)：半字长二进制参数，制定资源数据的长度。

MAXLIFETIME(cvda)：指定 CICS 自动释放资源的条件，cvda 可以选择 UOW，资源在本 UOW 结束后自动释放；还可选择 TASK，资源在本 TASK 结束时自动释放。

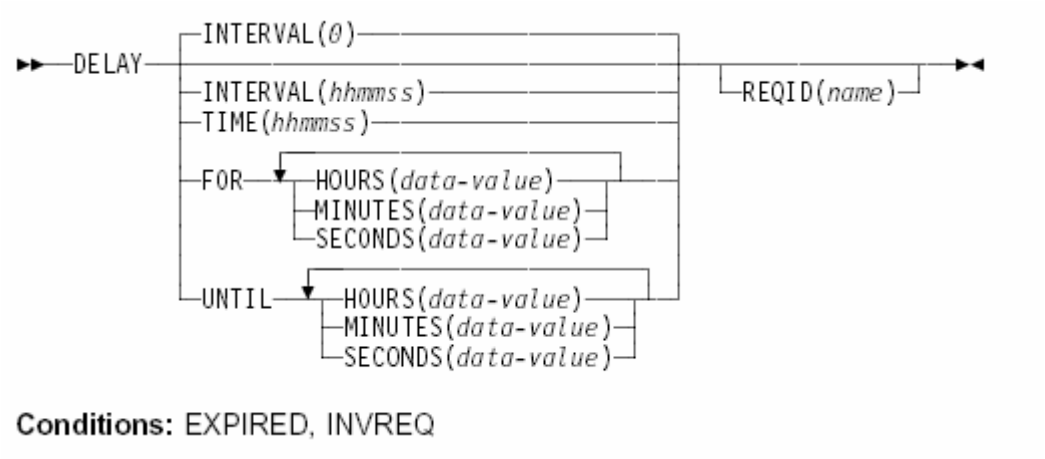
TASK：MAXLIFETIME(TASK)的另一种写法。

UOW：MAXLIFETIME(UOW）的另一种写法。

异常：

INVREQ：非法请求。

LENERR：长度错误。



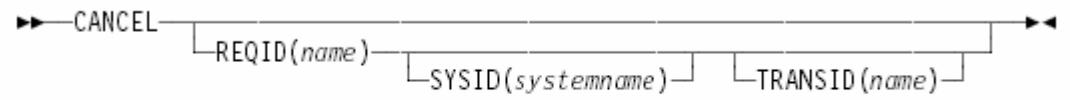
DELAY 命令用来使程序在指定的时刻暂停一定的时间，默认情况时立即暂停。DELAY 命令可以指定一个 REQID，其他的程序可以在暂停结束之前使用 CANCEL 命令来终止它的暂停。

命令选项：

INTERVAL(hhmmss)：	指定暂停在多长时间后开始。
TIME(hhmmss)：	指定暂停多长时间
FOR：	以另一中形式指定暂停时间
UNTIL：	指明暂停的结束时间。
REQID(name)：	指明一个 1-8 字节长的 REQID。

异常：

EXPIRED：	指定的时间已经过去了
INVREQ：	非法请求，时间格式不对 。



Conditions: ISCINVREQ, NOTAUTH, NOTFND, SYSIDERR

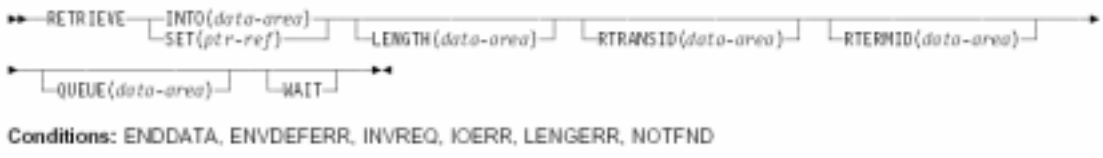
CANCEL 命令用来取消一个尚未完成的 DELAY, POST 或 START 命令。

命令选项：

REQID(name)：	指定一个 1-8 字节的 REQID，这个 REQID 必须是唯一的，与被 CANCEL 的命令所得到的 REQID 一致。
SYSID(systemname)：	指定一被 CANCEL 命令所在的远程系统名称
TRANSID(name)：	指明被 CANCEL 的交易 ID。如果此交易是远程交易，即使不指定 SYSID，处理也会自动转到远程执行。

异常

ISCINVREQ：	远程系统上操作错误
NOTAUTH：	用户权限不足
NOTFND：	没有找到这个 REQID
SYSIDERR：	远程系统名错



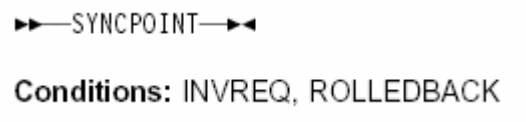
RETRIVE 命令用来接收 START 命令传过来的数据，这是接收 START 传输数据的唯一方法。

命令选项：

INTO(data-area)：	指定接收数据的区域。
LENGTH(data-area)：	半字长二进制参数，指定接收数据数据区的长度。若实际数据大于此值，数据被截断，并引发 LENGERR 异常。
SET(ptr-ref)：	指定接收数据的指针
QUEUE(data-area)：	指定 RETRIVE 的程序能去访问的 TSQ 名字。
RTERMID(data-area)：	4 字节的数据区去接收 START 发出的 RTERMID
RTRANSID(data-area)：	4 字节的数据区去接收 START 发出的 RTRANSID
WAIT：	指明当数据已被读取时等待直到有新数据

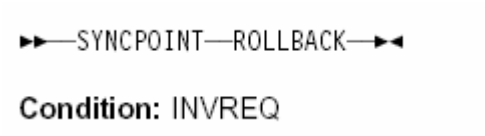
异常：

ENDDATA：	已无数据
ENVDEFERR：	发出 RETRIVE 的交易不是被 START 命令启动的。
INVREQ：	非法请求
IOERR：	I/O 错误
LENGERR：	长度错误
NOTFND：	没有数据



SYNCPOINT 命令把一个 TASK 分为一个以上的 UOW，把前一个 UOW 的内容提交。程序结束时系统会自动下 SYNCPOINT 命令，

异常：
INVREQ：非法请求。
ROLLEDBACK：COMMIT 失败。



SYNCPOINT ROLLBACK 命令恢复从上一个 SYNCPOINT 到当前的所有资源改变。若程序修改了远程资源，系统将远程资源一并恢复。

第八章　系统提供的交易

CICS 系统本身提供了许多系统交易，通过这些交易能够完成像资源配置、管理，程序追踪调试等很多功能，这里对常用的一些交易作简要的介绍：

- CEBR
- CECI
- CEDA
- CEDF
- CEMT
- CESN
- CESF

8-1 CEBR

```
 CEBR  TSQ ABIS.LOG          SYSID DVP1 REC      1 OF    128     COL      1 OF     67
 ENTER COMMAND ===>
      ***************************  TOP OF QUEUE  ******************************
00001 20040729 111959 112127 ::0009803 SCOS052 4701 T 02-Ip00 Ip00 E Tout
00002 20040729 114525 114525 ::0009887 SCOS052 4701   02-Ip00 Ip00 N
00003 20040729 114654 114735 ::0009934 SCOS052 4701 T 02-Ip00 Ip00 N
00004 20040729 122631 122631 ::0010005 SCOS052 4701   02-Ip00 Ip00 N
00005 20040729 123018 123018 ::0010008 SCOS052 4701   02-Ip00 Ip00 N
00006 20040729 124831 124831 ::0010020 SCOS052 4701   02-Ip00 Ip00 N
00007 20040729 124856 124856 ::0010022 SCOS052 4701   02-Ip00 Ip00 N
00008 20040729 124941 124941 ::0010024 SCOS052 4701   02-Ip00 Ip00 N
00009 20040729 125007 125007 ::0010025 SCOS052 4701   02-Ip00 Ip00 N
00010 20040729 125105 125105 ::0010026 SCOS052 4701   02-Ip00 Ip00 N
00011 20040729 125124 125124 ::0010027 SCOS052 4701   02-Ip00 Ip00 N
00012 20040729 125152 125152 ::0010028 SCOS052 4701   02-Ip00 Ip00 N
00013 20040729 125451 125451 ::0010031 SCOS052 4701   02-Ip00 Ip00 E 1569
00014 20040729 130044 130044 ::0010032 SCOS052 4701   02-Ip00 Ip00 N
00015 20040729 130144 130144 ::0010034 SCOS052 4701   02-Ip00 Ip00 N
00016 20040729 130222 130222 ::0010035 SCOS052 4701   02-Ip00 Ip00 N

 PF1 : HELP               PF2 : SWITCH HEX/CHAR     PF3 : TERMINATE BROWSE
 PF4 : VIEW TOP           PF5 : VIEW BOTTOM         PF6 : REPEAT LAST FIND
 PF7 : SCROLL BACK HALF   PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
 PF10: SCROLL BACK FULL   PF11: SCROLL FORWARD FULL PF12: UNDEFINED
```

CEBR 交易用来浏览和删除 TS QUEUE 数据。CEBR 有两种启动方式：第一种是直接在 CICS 终端上打 CEBR QUEUE-NAME，如键入“CEBR ABIS.LOG”就进入上面的画面，显示“ABIS.LOG”队列的内容；另一种方式是在 CEDF 调试程序时，进入内存显示/编辑画面时按 PF2 键，进入 CEBR，处理的队列名是“CEBR”+当前终端号。在 CEBR 界面里，可以使用 PF7/8 上下翻屏，PF10/11 左右翻屏，PF2 显示二进制文件，按 PF1 键显示帮助菜单，显示可以在“COMMAND ===>”输入的命令的解释：

Line 行号：　　直接跳到所给的行号显示，如“LINE 100”显示当前第 100 行记录。

Column 列号：　　把所给的列号置于最左侧显示，如“C 80”屏幕显示从 80 列开始的数据。

Top：　　跳到第一条记录

Bottom：　　跳到第一条记录

TERMinal terminal-id：看另一个终端的临时队列

Queue TSQ 名：　　查看另一个名字的 TSQ

Sysid shared/remote sysid ：查看另一个 REGION 的 TSQ

Put TDQ 名：　　把当前 TSQ 里的数据考到一个 TDQ 里去。

Get TDQ 名：　　把一个 TDQ 里的数据考到当前 TSQ 里来

PURGE：　　清除当前 TSQ

相关命令：CEMT INQ TSQ 显示当前有数据的 TSQ 列表。

```
 STATUS:  ENTER ONE OF THE FOLLOWING

ABend        DELAy        FREE         POSt         RETUrn       SYncpoint
ACquire      DELETE       FREEMain     PURge        REWInd       TEst
ADD          DELETEQ      GDs          PUSh         REWRite      TRace
ADDRess      DEQ          GET          PUT          ROute        UNlock
ALlocate     DISAble      GETMain      Query        RUn          UPdate
ASKtime      DISCard      GETNext      READ         SENd         Verify
ASSign       DOcument     Handle       READNext     SET          WAIT
BIf          DUmp         IGnore       READPrev     SIGNOFf      WAITCics
BUild        ENAble       INquire      READQ        SIGNON       WEb
CAncel       ENDBR        ISsue        RECeive      SPOOLClose   WRITE
CHAnge       ENDBROwse    Journal      RELease      SPOOLOpen    WRITEQ
CHEck        ENQ          LInk         REMove       SPOOLRead    Xctl
COLlect      ENTer        LOad         RESET        SPOOLWrite
CONNect      EXtract      Monitor      RESETBr      START
CONVerse     FEpi         PErform      RESUme       STARTBR
CReate       FORCe        POInt        RESYnc       STARTBROwse
DEFine       FORMattime   POP          RETRieve     SUspend



PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER             9 MSG
```

CECI 是 CICS 命令的输入界面，可以在 CECI 中直接输入 EXEC CICS 。。。所用的命令。使用方法可以是逐段地输入命令，根据提示完成整个命令，也可以一次输入整个命令如：“READ FILE(SAVING) INTO(B) RIDFLD(0000001000)”。

CECI 中功能键使用如下：

PF01 显示帮助
PF02 二进制显示
PF03 结束 CECI
PF04 显示 EIB
PF05 变量操作（见后述）
PF06 显示用户输出界面
PF07 上翻半屏
PF08 下翻半屏
PF09 显示命令执行信息，如查看异常情况
PF10 上翻整屏
PF11 下翻整屏

8-2　CECI（续页 1）

CECI 中执行的命令中经常要使用到变量。这些变量要预先定义。进入 CECI 后，PF5 键进入变量操作界面，可以增加自己要用的变量，如下：

```
 VARIABLES   LENGTH   DATA
&DFHC        +00016   THIS IS A SAMPLE
&DFHW        +00046   EXEC CICS WRITEQ QUEUE('"CI0053') FROM(&DFHC)
&DFHR        +00045   EXEC CICS READQ QUEUE('"CI0053') INTO(&DFHC)
a             20
b             100

PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER             9 MSG
```

回车，这些变量就已分配：

```
 VARIABLES   LENGTH   DATA
&DFHC        +00016   THIS IS A SAMPLE
&DFHW        +00046   EXEC CICS WRITEQ QUEUE('"CI0053') FROM(&DFHC)
&DFHR        +00045   EXEC CICS READQ QUEUE('"CI0053') INTO(&DFHC)
&A           +00020
&B           +00100

PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER             9 MSG
```

再回车回到前一画面，然后就可以输入 CICS 命令，如：

```
  READ FILE(SAVING) INTO(B) RIDFLD(0000001000)
  STATUS:  ENTER ONE OF THE FOLLOWING
ABend        DELAy        FREE         POSt         RETUrn       SYncpoint
 ACquire      DELETE       FREEMain     PURge        REWInd       TEst
 ADD          DELETEQ      GDs          PUSh         REWRite      TRace
 ADDRess      DEQ          GET          PUT          ROute        UNlock
 ALlocate     DISAble      GETMain      Query        RUn          UPdate
 ASKtime      DISCard      GETNext      READ         SENd         Verify
 ASSign       DOcument     Handle       READNext     SET          WAIT
 BIf          DUmp         IGnore       READPrev     SIGNOFf      WAITCics
 BUild        ENAble       INquire      READQ        SIGNON       WEb
 CAncel       ENDBR        ISsue        RECeive      SPOOLClose   WRITE
 CHAnge       ENDBROwse    Journal      RELease      SPOOLOpen    WRITEQ
 CHEck        ENQ          LInk         REMove       SPOOLRead    Xctl
 COLlect      ENTer        LOad         RESET        SPOOLWrite
 CONNect      EXtract      Monitor      RESETBr      START
 CONVerse     FEpi         PErform      RESUme       STARTBR
 CReate       FORCe        POInt        RESYnc       STARTBROwse
 DEFine       FORMattime   POP          RETRieve     SUspend
PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER             9 MSG
```

回车就可以看到执行结果：

```
 READ FILE(SAVING) INTO(B) RIDFLD(0000001000)
 STATUS:  COMMAND EXECUTION COMPLETE                          NAME=
  EXEC CICS  READ
   File( 'SAVING  ' )
   < SYsid() >
   ( SEt()
     | Into( '0000001000AAAAAAAAAAAAAAAAAAAAAAAAAAAA            ' ... ) )
   < Length( +00080 ) >
   RIdfld( '0000001000' )
   < Keylength() < GEneric > >
   < RBa | RRn | DEBRec | DEBKey >
   < GTeq | Equal >
   < UNcommitted | Consistent | REpeatable | UPdate < Token() > >
   < Nosuspend >

  RESPONSE: NORMAL               EIBRESP=+0000000000 EIBRESP2=+0000000000
PF 1 HELP 2 HEX 3 END 4 EIB 5 VAR 6 USER 7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

8-3 CEDA

CEDA 是联机资源定义交易，在前面的内容里我们已经初步使用了它的一些功能，实际上 CEDA 有十分强大的功能。直接键入 CEDA 命令将会看到其引导画面：

```
  ENTER ONE OF THE FOLLOWING
 ADd
 ALter
 APpend
 CHeck
 COpy
 DEFine
 DELete
 DIsplay
 Expand
 Install
 Lock
 Move
 REMove
 REName
 UNlock
 USerdefine
 View
                                                    SYSID=DVP1
APPLID=DVPCICS1
PF 1 HELP       3 END              6 CRSR             9 MSG             12 CNCL
```

在 CEDA 中我们可以：

ADd: 增加一个 LIST，或向 LIST 中增加一个 GROUP
ALter: 修改现有资源的一些属性
APpend: 把一个 LIST 里定义的资源拷贝到另一个 LIST 里去。
CHeck: 检查在一个 LIST 或 GROUP 里定义的资源与另一个里的一致。
COpy: 复制一个或多个资源定义.
DEFine: 定义一个新资源。
DELete: 删除一个资源.
DIsplay: 显示一个 LIST 或 GROUP 里定义的所有资源.
Expand: 展开一个 GROUP 里定义的资源.
Install: 动态安装资源,使资源变为可用.
Lock: 限制只能某一个用户才能修改或删除特定 LIST 或 GROUP 里的资源
Move: 把某资源从一个 GROUP 移动到另一个 GROUP 里去.
REMove: 从 LIST 里删除一个 GROUP.
REName: 修改一项资源的名称.
UNlock: 释放某个 CEDA LOCK 命令做的限制
USerdefine: 不使用系统 DEFINE 命令提供的各参数默认值,使用 USERDEFINE 命令可以用自己定义的默认值来定义资源
View: 查看资源定义.
其中每个命令下又有复杂的选项。

8-3　CEDA（续页 1）

我们可以一次输入全部的命令然后执行也可以根据画面提示逐步完善命令，如下面就是 CEDA DEF 后的提示画面，可以继续进行各种资源的定义：

```
  DEF
  ENTER ONE OF THE FOLLOWING

 Connection   TCpipservice
 DB2Conn      TDqueue
 DB2Entry     TErminal
 DB2Tran      TRANClass
 DOctemplate  TRANSaction
 Enqmodel     TSmodel
 File         TYpeterm
 Journalmodel
 Lsrpool
 Mapset
 PARTItionset
 PARTNer
 PROCesstype
 PROFile
 PROGram
 Requestmodel
 Sessions

                                                    SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP       3 END             6 CRSR             9 MSG             12 CNCL
```

也可以一次输入全部命令，如：“CEDA VIEW TRANS(TRN1) GROUP(TESTGP)”　第一页

```
  VIEW TRANS(TRN1) GROUP(TESTGP)
  OBJECT CHARACTERISTICS                                      CICS RELEASE = 0530
   CEDA  View TRANSaction( TRN1 )
    TRANSaction    : TRN1
    Group          : TESTGP
    DEscription    :
    PROGram        : SAMPLE1
    TWasize        : 00000             0-32767
    PROFile        : DFHCICST
    PArtitionset   :
    STAtus         : Enabled           Enabled | Disabled
    PRIMedsize     : 00000             0-65520
    TASKDATALoc    : Below             Below | Any
    TASKDATAKey    : User              User | Cics
    STOrageclear   : No                No | Yes
    RUnaway        : System            System | 0 | 500-2700000
    SHutdown       : Disabled          Disabled | Enabled
    ISolate        : Yes               Yes | No
    Brexit         :
 + REMOTE ATTRIBUTES

                                                    SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP 2 COM 3 END             6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

按 PF11 翻页：

```
  VIEW TRANS(TRN1) GROUP(TESTGP)
  OBJECT CHARACTERISTICS                                  CICS RELEASE = 0530
   CEDA  View TRANSaction( TRN1 )
    TRANSaction    : TRN1
    Group          : TESTGP
    DEscription    :
    PROGram        : SAMPLE1
    TWasize        : 00000             0-32767
    PROFile        : DFHCICST
    PArtitionset   :
    STAtus         : Enabled           Enabled | Disabled
    PRIMedsize     : 00000             0-65520
    TASKDATALoc    : Below             Below | Any
    TASKDATAKey    : User              User | Cics
    STOrageclear   : No                No | Yes
    RUnaway        : System            System | 0 | 500-2700000
    SHutdown       : Disabled          Disabled | Enabled
    ISolate        : Yes               Yes | No
    Brexit         :
 + REMOTE ATTRIBUTES
                                                       SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

继续翻页：

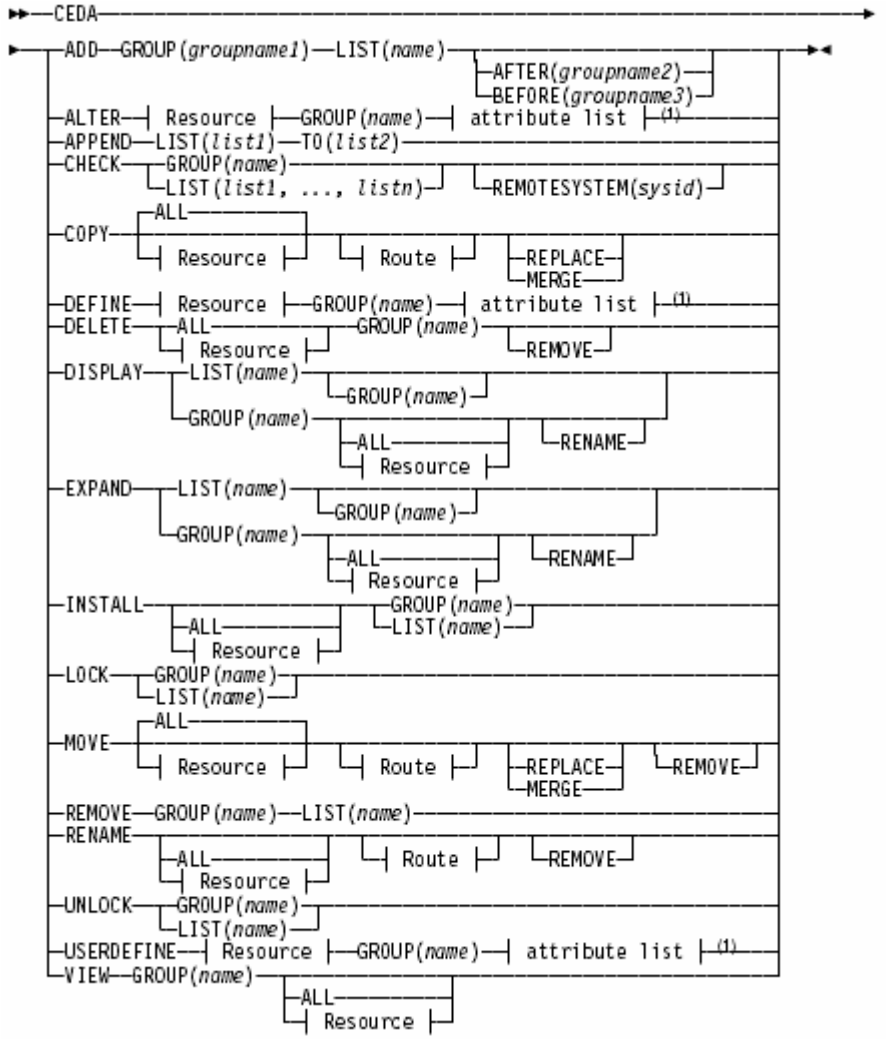
```
  VIEW TRANS(TRN1) GROUP(TESTGP)
  OBJECT CHARACTERISTICS                                  CICS RELEASE = 0530
   CEDA  View TRANSaction( TRN1 )
 +                 :
                   :
   RECOVERY
    DTimout        : No                No | 1-6800
    RESTart        : No                No | Yes
    SPurge         : No                No | Yes
    TPUrge         : No                No | Yes
    DUmp           : Yes               Yes | No
    TRACe          : Yes               Yes | No
    COnfdata       : No                No | Yes
   INDOUBT ATTRIBUTES
    ACtion         : Backout           Backout | Commit
    WAIT           : Yes               Yes | No
    WAITTime       : 00 , 00 , 00      0-99 (Days,Hours,Mins)
    INdoubt        : Backout           Backout | Commit | Wait
   SECURITY
 +  RESSec         : No                No | Yes
                                                       SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

8-3　CEDA（续页 3）

最后一页：

```
  VIEW TRANS(TRN1) GROUP(TESTGP)
  OBJECT CHARACTERISTICS                                    CICS RELEASE = 0530
   CEDA  View TRANSaction( TRN1 )
 +  CMdsec          : No                 No | Yes
    Extsec          : No                 No | Yes
    TRANSec         : 01                 1-64
    RS1             : 00                 0-24 | Public




                                                          SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP 2 COM 3 END              6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

CEDA 的全部选项：



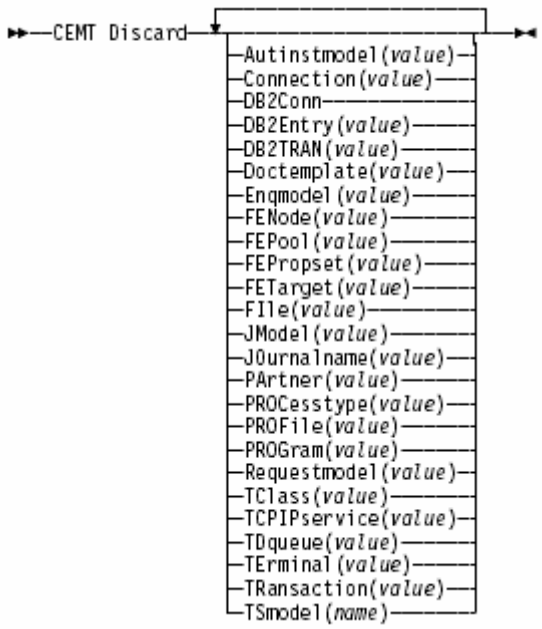
CEDF 是 CICS 提供的调试交易，具体使用方法见第二章的介绍。

CEMT 提共 MASTER TERMINAL 功能，使用户能动态地控制 CICS 系统。CEMT 提供以下四种主要操作：

```
  STATUS:  ENTER ONE OF THE FOLLOWING

 Discard
 Inquire
 Perform
 Set









                                                        SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP       3 END       5 VAR                      9 MSG
```

DISCARD 操作：

使用 DISCARD 操作你可以删除一个活动 CICS 系统上的某项资源定义，DISCARD 命令并不影响 CICS 系统定义（CSD）的内容，也就是当 CICS 重新启动时，CEMT 作的这些操作会被冲掉。DISCARD 命令语法如下：



INQUIRE 命令：

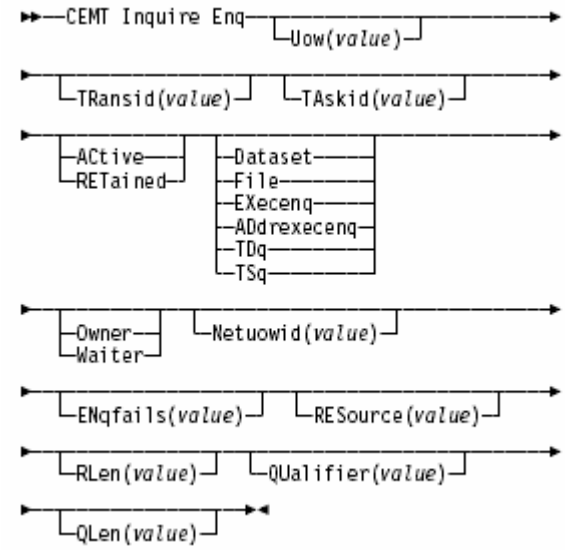
INQUIRE 命令是 CEMT 最常用的命令，用来查看系统动态信息，可以进行以下查询：

```
  INQ
  STATUS:  ENTER ONE OF THE FOLLOWING OR HIT ENTER FOR DEFAULT

 AUTInstmodel FENode       PROGram      TSQueue
 AUTOinstall  FEPOol       REquestmodel UOW
 AUXtrace     FEPRopset    RRms         UOWDsnfail
 CFdtpool     FETarget     STAtistics   UOWLink
 COnnection   FIle         STReamname   Vtam
 DB2Conn      Gtftrace     SYDumpcode   Web
 DB2Entry     INttrace     SYStem
 DB2Tran      IRc          TAsk
 DEletshipped JModel       TCLass
 DOctemplate  JOurnalname  TCPIP
 DSAs         Line         TCPIPService
 DSName       MODename     TDqueue
 DUmpds       MONitor      TErminal
 ENQ          Netname      TRAnsaction
 ENQModel     PArtner      TRDumpcode
 EXci         PROCesstype  TSModel
 FEConnection PROFile      TSPool
                                                    SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP       3 END       5 VAR                  9 MSG
```

现仅就常用的几中查询进行介绍：

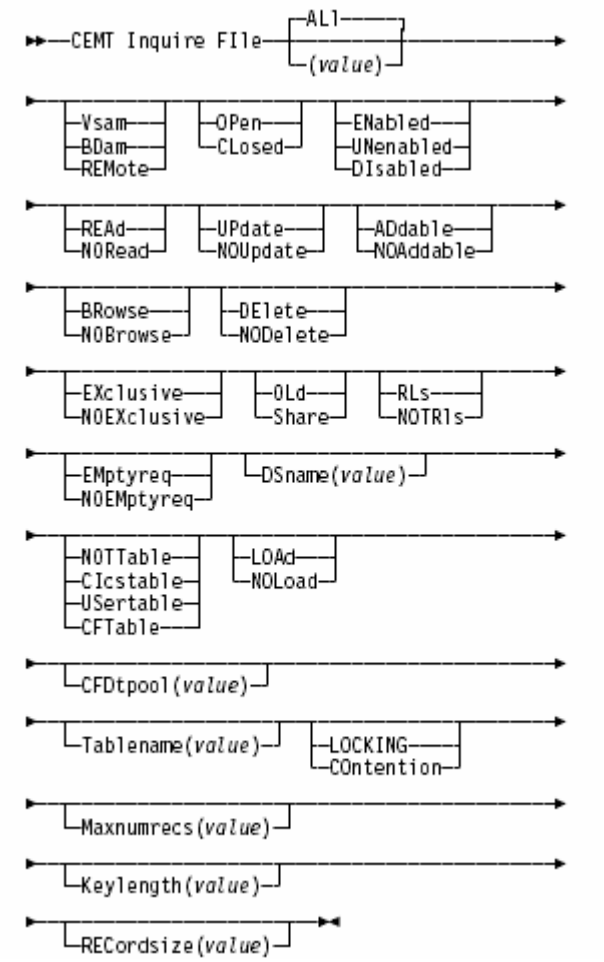
CEMT INQUIRE ENQ：

查询系统内的 ENQ 情况，语法如下：



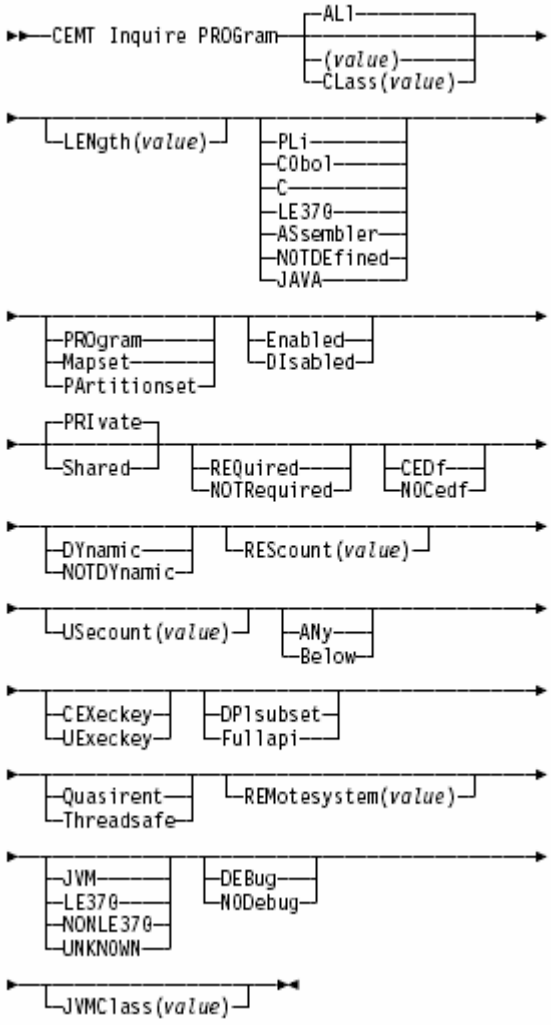
CEMT INQUIRE FILE：

查询文件资源情况，语法如下：



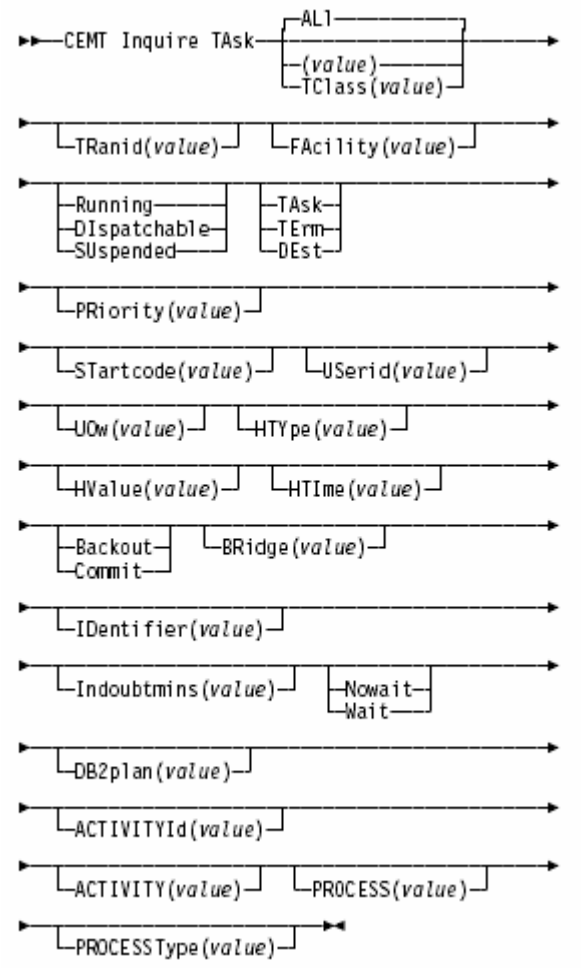
CEMT INQUIRE PROGRAM：

查询 CICS 程序资源情况，语法如下：



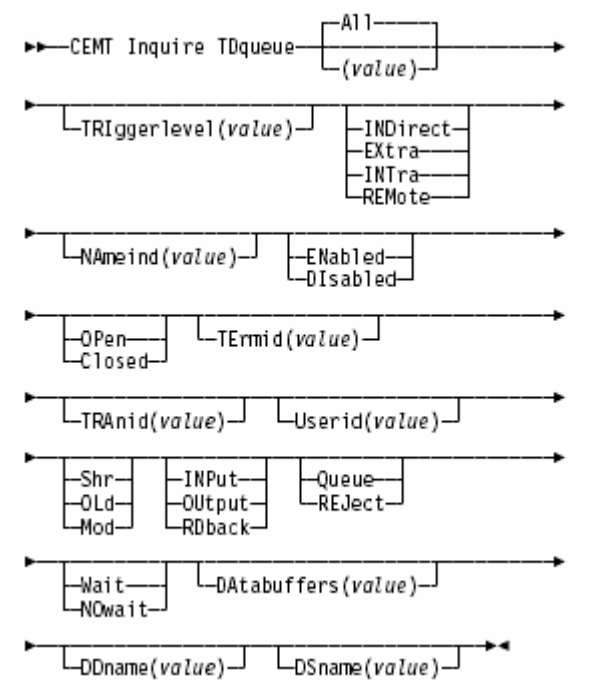
CEMT INQUIRE TASK：

查询 CICS 系统当前正运行的 TASK 情况，语法如下：



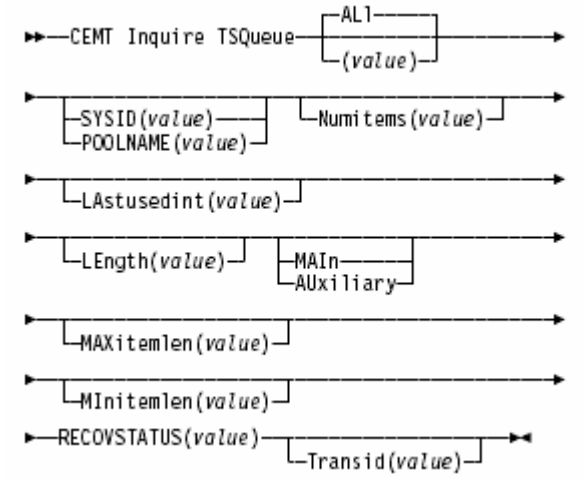
CEMT INQUIRE TDQUEUE：

查询 TDQ 资源情况，语法如下：



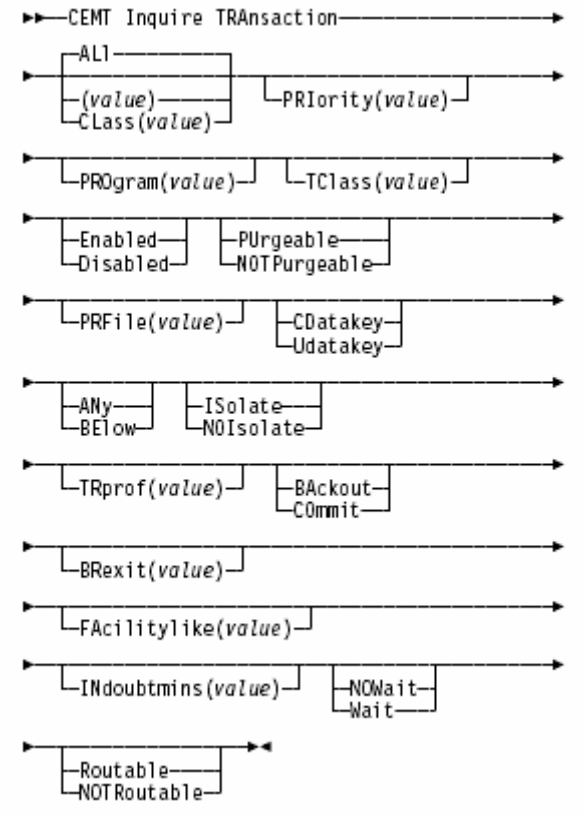
CEMT INQUIRE TSQUEUE：

查询 TSQ 资源情况，语法如下：



CEMT INQUIRE TRANSACTION：

查询 TRANSACTION 资源情况，语法如下：



CEMT PERFORM 命令：

```
  PERFORM
  STATUS:  ENTER ONE OF THE FOLLOWING

 DEletshipped
 DUmp
 Endaffinity
 Reset
 SEcurity
 SHUTdown
 SNap
 STatistics




                                                        SYSID=DVP1
APPLID=DVPCICS1

 PF 1 HELP       3 END       5 VAR                      9 MSG
```

命令说明：

DEletshipped：	使 CICS 超时清除机制生效(PERFORM DELETSHIPPED)
DUmp：	生成 CICS DUMP(PERFORM DUMP or PERFORM SNAP)
Endaffinity：	关闭一个 CICS 连接
Reset：	使 CICS 重新与西统时间对表(PERFORM RESET)
SEcurity：	重建 RACF 资源 PROFILE (PERFORM SECURITY REBUILD)
SHUTdown：	关闭 CICS (PERFORM SHUTDOWN)
STatistics：	纪录 CICS 统计信息(PERFORM STATISTICS RECORD).

8-5　CEMT（续页 8）

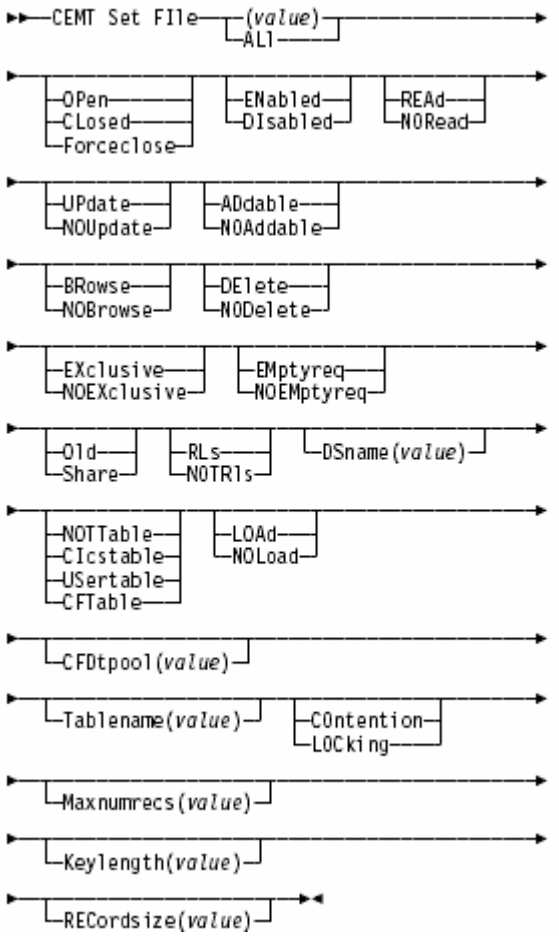
SET 命令：

SET 命令用来对系统的各种资源状态作出改变，可以操作以下资源：

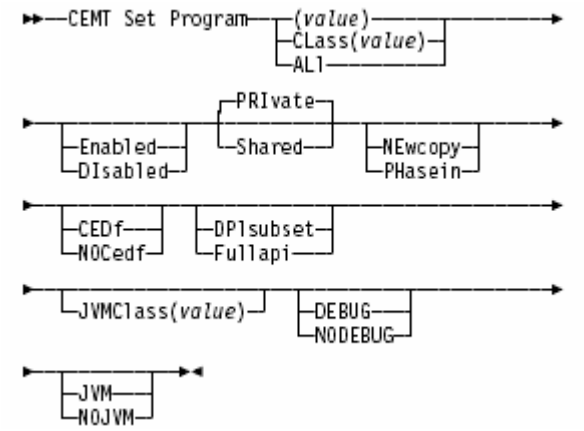
```
  SET
  STATUS:  ENTER ONE OF THE FOLLOWING OR HIT ENTER FOR DEFAULT
AUToinstall  INttrace     TDqueue
 AUXtrace     IRc          TErminal
 Connection   Journalname  TRAnsaction
 DB2Conn      Line         TRDumpcode
 DB2Entry     MODename     TSqueue
 DB2Tran      MONitor      UOW
 DEletshipped Netname      UOWLink
 DSAs         PROCesstype  Vtam
 DSName       PROGram      Web
 DUmpds       Requestmodel
 Enqmodel     STatistics
 FEConnection SYDumpcode
 FENode       SYStem
 FEPool       TAsk
 FETarget     TCLass
 FIle         TCPIP
 Gtftrace     TCPIPService
                                                    SYSID=DVP1
APPLID=DVPCICS1
PF 1 HELP       3 END       5 VAR                   9 MSG
```

下面仅就一些常用选项进行介绍：

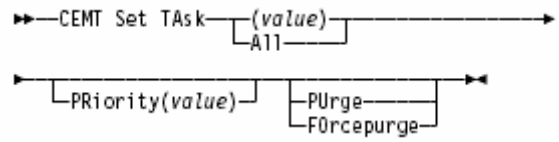
CEMT SET FILE：设置文件资源的状态，语法如下：



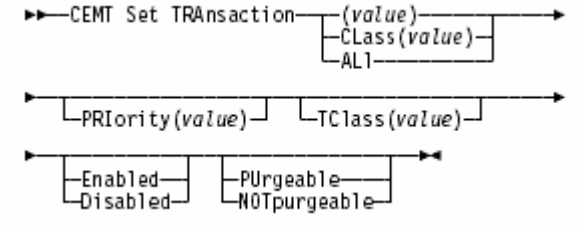
CEMT SET PROGRAM：设置程序资源的状态，语法如下：



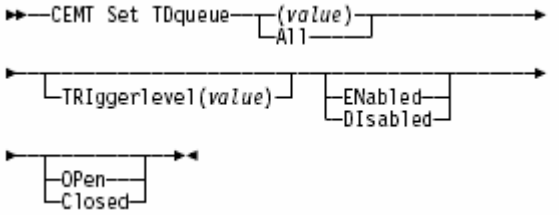
CEMT SET TASK：设置当前正在运行的交易的状态，语法如下：



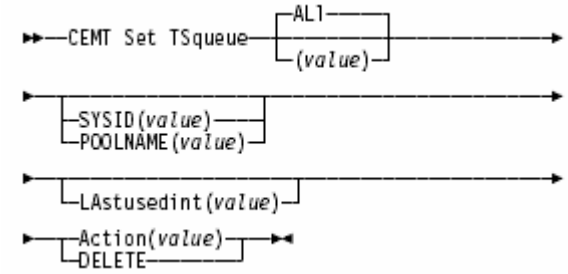
CEMT SET TRANSACTION：设置 TRANSACTION 资源的状态，语法如下：



CEMT SET TDQUEUE：设置 TDQ 资源的状态，语法如下：



CEMT SET TSQUEUE：设置 TSQ 资源的状态，语法如下：



```
                              Signon to CICS                    APPLID DVPCICS1

WELCOME TO PRODUCTION CICS REGION !




Type your userid and password, then press ENTER:

         Userid . . . .              Groupid . . .
         Password . . .
         Language . . .

     New Password . . .









DFHCE3520 Please type your userid.
F3=Exit
```

CICS 的签到交易，签到后将以签到用户的身份接受各种权限检查。

8-7 CESF

CESF 是 CICS 的签退交易，用法：CESF LOGOFF 或 CESF GOODNIGHT

练习 6

练习

练习一

练习目的：

初步编制一个简单的 CICS 程序，熟悉 CICS 程序的编译、运行和调试的过程。

练习要求：

根据第二章的介绍，编制一个 CICS 程序，它接受 “交易码 XXXX”的输入，其中 XXXX 是一个 EIB 变量名称。程序根据输入的具体变量名称，把指定的 EIB 变量显示出来。

在 ISPF 中编辑完成程序，根据课程的介绍编写自己的编译 PROC 和 JCL，使用 JCL 对程序进行编译，根据编译结果改正程序中的语法错误。

在 CICS 终端为自己的程序定义一个交易，运行交易，调试改正程序可能的逻辑错误，并练习使用 CEDF 来对程序进行追踪。

练习二

练习目的：

- ✓ 熟悉 MAPSET 的设计与定义
- ✓ 熟悉 MAPSET 宏的编译
- ✓ 熟悉使用 BMS 程序的编写
- ✓ 熟悉使用 BMS 程序的编译与执行

练习要求：

仿照银行取款业务设计一组屏幕映像，包括一个输入画面，画面中有帐号、密码、取款金额和备注字段，和一个输出画面，包括姓名、帐号、余额字段。使用 DFHMSD 宏来编写 MAPSET 的定义，然后按照第三章的介绍，编制 JCL 编译宏，生成物理映像和字符映像。

编写 COBOL 程序使用如上的 MAP 来接收数据并显示结果，使用伪会话交易，编译生成可执行代码

在 CICS 中定义并安装此交易及其他所需资源，调试运行此程序。

练习三

练习目的：

- ✓ 熟悉 VSAM 文件的定义
- ✓ 熟悉在程序中使用 VSAM 文件

练习要求：

仿照银行业务，设计一个 KSDS　VSAM 文件存放客户存款信息，包括帐号、姓名、密码、余额字段，其中帐号字段是 KEY。使用 DITTO 或其他工具增加部分记录进取。在练习二的基础上，把程序增加数据处理功能，根据输入的信息，到 VSAM 文件中找到这条记录，比对密码，对账户余额扣账，然后写回 VSAM 文件，向终端输出结果信息。

选做：

仿照提款交易，编写开户交易、存款交易和销户交易程序。

练习目的：

- ✓ 程序控制命令的使用
- ✓ 熟悉内存控制命令的使用

练习要求：

在练习三的基础上，把接收/发送屏幕显示和文件数据处理的逻辑分离，写成两个程序，由一个调用另一个进行处理。尝试使用 LINK 和 XCTL 两种方式进行处理；数据传输方式也使用 COMMAREA 和 INPUTMSG 两种方法。然后再尝试在程序中使用 GETMAIN 和 FREEMAIN 来学习内存控制命令。

练习目的：

- ✓ CICS 队列的使用

练习要求：

在练习四的基础上，改变程序间的数据传输方式，改用调用程序写 TSQ，被调用程序从 TSQ 中读出需要的数据，再进行处理，处理结果通过另一个 TSQ 返回给调用方。

练习目的：

熟悉第八章介绍的各个 CICS 提供的交易：

- ✓ CEBR
- ✓ CECI
- ✓ CEDA
- ✓ CEDF
- ✓ CEMT
- ✓ CESN
- ✓ CESF

练习要求：

依次使用上述的几个交易，熟悉并使用交易里的各个分支功能。