XMODEM/YMODEM协议参考

描述该协议的一系列文档汇编

XMODEM和

YMODEM

文件传输协议

此文件于1987年4月8日格式化。
此文件于2024年8月22日被翻译成中文

由ChuckForsberg编辑
由miuser翻译

向ChuckForsberg提问

奥门科技有限公司

高可靠性软件

17505-Sauvie岛公路

波特兰俄勒冈州37231

语音：503-621-3406

语音调制解（TeleGodzilla）

503-621-3746

速度 13200,2400,1200,300

Compuserve：70007,2304
Genie：CAF
UUCP:...!tektronix!reed!omen!caf

目录

图表表

没耐心的读者建议直接跳到图13，这里描述了完整的一般传输过程

1. 巴别塔

一个“YMODEM通天塔”降临到微型计算机社区，带来了混乱、挫败感、膨胀的电话账单和浪费的人时。遗憾的是，我（ChuckForsberg）对此混乱部分负有责任。

作为20世纪80年代初批处理和1kXMODEM扩展的作者，我假设早期版本的文档读者会根据他们的编程技能和计算环境尽可能实现YMODEM协议。这证明是一个过于天真的假设，因为受到竞争压力的程序员尽可能少地实现了YMODEM。有些人采取了YMODEM中吸引他们的任何部分，将其应用于MODEM7批处理、Telink、XMODEM或其他，并称之为YMODEM。

杰夫·加伯斯（Crosstalk包开发总监）一语道破天机：“随着协议进入公共领域，任何想摆弄它们的人都可以继续。”包含从YMODEM.DOC派生出的修改示例的文档增加了混乱。在一个例子中，YMODEM.DOC的图1的标题已从“1024字节数据包”变为“YMODEM/CRC文件传输协议”。一个文档中展示的XMODEM和YMODEM示例中没有一个是对的。

为了结束这种混淆，我们必须清楚地说明YMODEM代表什么，正如WardChristensen在1 385年创造这个术语时所定义的那样。

对于那些阅读、理解和尊重Ward对YMODEM定义的大多数人，我为带来的不便表示歉意。

1.1定义

ARCARC 是一个程序，它将一个或多个文件压缩成一个档案，并从这些档案中提取文件。

XMODEM指的是由WardChristensen在1977年推出的MODEM.ASM程序所引入的文件传输礼仪。XMODEM这个名字来源于KeithPetersen的XMODEM.ASM程序，它是为远程CP/M（RCPM）系统对MODEM.ASM程序进行改编的。它也被称为MODEM或MODEM2协议。一些不了解MODEM7不寻常的批处理文件模式的用户称其为MODEM7。其他别名包括“CP/M用户组”和“TERMⅡFTP 3”。XMODEM之所以流行，部分是因为其独特性，部分是因为媒体对使用“XMODEM”命令访问的公告板和RCPM系统的兴趣。这个协议因为其通用性、简单性和合理的性能而被每个严肃的通信程序所支持。

XMODEM/CRC用两个字节的循环冗余校验（CRC）替换了XMODEM的1字节校验和。

---

1 . 第C/12页，《PC周刊》1387年7月12日

检查（CRC-16），提供现代错误检测保护。

XMODEM-1k 指的是使用1024字节数据块的XMODEM/CRC协议。

YMODEM指的是如以下所述的XMODEM/CRC（可选1k块）协议，具有批量传输功能。

真YMODEM™为了解决YMODEM巴别塔问题，OmenTechnology已经将 真YMODEM™ 一词注册为商标，以代表本文件中描述的完整YMODEM协议，包括在块0中传输的路径名、长度和修改日期。请联系OmenTechnology了解关于真YMODEM™合规性认证的程序。

ZMODEM使用熟悉的XMODEM/CRC和YMODEM技术，在一种新的协议中提供适合当代数据通信的可靠性、吞吐量、文件管理和用户便利性。

ZOO，类似于ARC，ZOO是一个将一个或多个文件压缩成“ZOO存档”的程序。 ZOO支持包括Unix和VMS在内的许多不同的操作系统。

2. YMODEM最低要求

所有声称支持YMODEM的程序必须满足以下最低要求：

发送程序应在块0中发送路径名（文件名）。

路径名应是一个以空字符终止（null terminated）的ASCII字符串，如下所述。

接收程序应使用此路径名作为接收文件的名称，除非明确覆盖。

发送程序应使用CRC-16响应"C"路径名拒绝，否则使用8位校验和。

接收程序必须接受其接收到的任何128字节和1024字节块的混合。

发送程序不得更改未确认块的长度。

在每个文件的末尾，发送程序应发送EOT（文件结束符）多达十次，直到它收到一个确认字符。（这是XMODEM规范的一部分。）

传输会话的结束应以空（空）路径名null (empty)表示。

不符合所有这些要求的程序不是YMODEM兼容的。

满足最低要求并不能保证在压力下可靠地传输文件。

3. 为什么选择YMODEM？

自从五年前开发以来，沃德·克里斯滕森调制解调器协议已经使各种计算机系统能够交换数据。几乎没有一个通信程序不至少声称支持这个协议。

计算、调制解调器和网络技术的最新进展已经揭示了原始协议中的许多弱点：

短块长度在使用时分系统、分组交换网络、卫星电路和缓冲（纠错）调制解调器时会导致吞吐量下降。
8位算术校验和其他方面允许线路损坏干扰可靠、准确的传输。

每条命令只能发送一个文件。文件名必须输入两次，第一次给发送程序，然后再次给接收程序。
传输的文件可能会累积多达127个无关字节。
文件的修改日期已丢失。
这些年来已经开发了许多其他协议，但至今没有一种协议取代XMODEM：

由于缺乏公共领域的文档和示例程序，诸如Blast、Relay等专有协议一直紧密依赖于其供应商的命运。
复杂性阻碍了BISYNC、SDLC、HDLC、X.25和X.PC协议的广泛应用。

性能妥协和复杂性限制了Kermit协议的普及，该协议旨在允许在敌对XMODEM的环境中进行文件传输。

XMODEM协议扩展和YMODEM批处理解决了一些这些弱点，同时保持了XMODEM的大部分简单性。YMODEM由公共领域的程序YAM(CP/M)、YAM(CP/M-86)、YAM(CCPM-86)、IMP(CP/M)、KMD(CP/M)、rz/sz(Unix、Xenix、VMS、伯克利Unix、Venix、Xenix、Coherent、IDRIS、Regulus)支持。

商业实现包括MIRROR和Professional-YAM。自1381年以来，已有支持这些扩展的通信程序在使用。

以下描述的1k块长度（XMODEM-1k）可以与YMODEM批处理协议结合使用，或者与XMODEM/CRC协议相同的单文件传输使用，除了对支持1k块的最小改动。

另一种扩展是YMODEM-g协议。YMODEM-g在与端到端纠错媒体（如X.PC和纠错调制解调器）一起使用时提供最大吞吐量，包括TeleBit、U.S.Robotics、Hayes、ElectronicVaults、Data Race等公司的3600bps设备。

---

[1]适用于IBMPC、XT、AT、Unix和Xenix

其他

为了完成这部著作，包括了WardChristensen原始协议文档和JohnByrns的CRC-16文档的编辑版本供参考。

关于MODEM或MODEM7协议的引用已更改为XMODEM，以适应方言。在澳大利亚，它正确地被称为Christensen协议。

一些先锋的信息

#:130340S0/通讯25-Apr-8518:38:47
Sb: 我的协议

发件人：

WardChristensen76703302 3,3021

收件人：所有人

请注意，文章2确实正确引用了我关于“不够稳健”等表述的内容。

这是一次我迅速拼凑起来的小改动，非常不计划（就像我做的所有事情一样），以满足与“某些人”沟通的个人需求。

仅仅是因为它在8/77年完成，并且我立即将其置于公共领域，才让它成为了现在的标准。

我认为是时候了

(1) 记录它；（人们打电话给我说“我的产品将要包括它——我可以引用什么”，或者“我正在写一篇关于它的论文，我在参考文献中应该写什么”）并且

(2) 提出对其的“增量扩展”，这可能正好采取ChuckForsberg的YAM协议的形式。他用C语言为CP/M编写了YAM并将其置于公共领域，并且为Unix3编写了一个批处理协议，称为rb和sb（接收批处理，发送批处理），这基本上是XMODEM的

(a) 包含文件名、日期、时间和大小的“记录0”

(b) 1K块大小选项

(c) CRC-16。

他进行了一些巧妙的编程来检测错误的ACK或EOT，但基本上还是保持原样。

那些建议我对协议进行重大修改的人，比如“全双工”、“多个待处理块”、“多个目的地”等等，他们不理解协议的这种不可思议的简单性正是它能够在如此多的机器和程序中存活至今的原因之一！

---

[1]排版外观已编辑
[2]信息世界 4月23日 第16页
[3]这些程序的VAX/VMS版本也可用。

考虑一下1377年左右的PC-NET小组——为了超越而进行记录——他们有一个协议，但它是“极其复杂的”，因为它试图“满足所有人的需求”——也就是说在7位系统上发送二进制文件等。我并不那么“仁慈”。我（强调>我<）有一个8位UART，所以“我的协议是一个8位协议”，我只会对那些受7位限制所阻碍的人说“抱歉”...
块大小：ChuckForsberg创建了我协议的扩展，称为YAM，它也通过他名为rb和sb的公共领域UNIX程序得到支持，分别是接收批量发送批量。它们巧妙地发送一个包含文件名、日期、时间和大小的“块0”。

不幸的是，它的UNIX风格，有点奇怪 如八进制数等。

但是，这是一种克服MODEM7引入的“回显名称字符”的不错方法。此外，Chuck使用CRC-16和可选的1K块。因此，记录0、1K和CRC使其成为“相当酷的新协议”，这与我的协议没有显著区别。

此外，有一个吸引人的名字——YMODEM。这意味着对某些人来说，它是“XMODEM之后的东西”，而对另一些人来说，它是Y(am)MODEM协议。我不想过多强调这一点——出于担心其他制造商可能会认为它是一个“竞争性”协议，而不是一个“非附属”协议。Chuck目前正在销售他增强版的CP/M-80C程序YAM，称之为ProfessionalYam，适用于PC——我现在就在使用它。非常酷！

32K捕获缓冲区，脚本，滚动，之前捕获的文本搜索，以及几乎一切内置命令-目录（按各种方式排序），XMODEM，YMODEM，KERMIT和ASCII文件上传/下载等。您可以编程使其与大多数系统“表现”-例如，当尝试为CIS输入一个数字时，它会检测来自调制解调器的“忙碌”字符串，并将不同的电话号码替换到拨号字符串中，然后回退以尝试。

文件长度，时间和文件模式是可选的。路径名如果需要，可能会单独发送文件长度。

4. XMODEM协议增强

本章讨论了对沃德·克里斯滕森1382年XMODEM协议描述文档的协议扩展。

原始文档建议在尝试10次后询问用户是否继续尝试或中止。大多数程序不再询问操作员是否希望继续重试。
几乎所有可纠正的错误都在前几次重传中被纠正。如果线路太差，十次尝试不足以解决问题，那么存在未检测到错误的风险。如果连接如此糟糕，最好是重新拨号以获得更好的连接，或者发送一个软盘。

4.1优雅终止

YAM和Professional-YAMX/YMODEM例程识别连续的两个CAN（十六进制18）字符序列，作为传输终止命令，且没有发生调制解调器错误（溢出、帧错误等）。当等待一个数据块的开始或等待对已发送数据块的确认时，会识别这个序列。检查连续的两个CAN字符可以减少因线路干扰而终止传输的次数。当YAM终止XMODEM、YMODEM或ZMODEM协议文件传输时，会发送八个CAN字符。然后Pro-YAM发送八个退格符来删除远程键盘输入缓冲区中的CAN字符，以防远程已经终止传输并等待键盘命令。

4.2 CRC-16 选项

XMODEM协议使用一个可选的两字符 CRC-16校验码，而不是原始协议及大多数商业实现所使用的单字符算术校验和。
CRC-16可以检测所有单比特和双比特错误，所有奇数个错误比特的错误，长度为16或更短的突发错误，以及所有17比特错误突发中的99.9969%，以及所有可能更长的错误突发中的99.9984%。相比之下，双比特错误或3比特或更长的突发错误可以绕过XMODEM协议的算术校验和。

XMODEM/CRC协议与XMODEM协议类似，不同之处在于接收方在请求第一个数据块时发送“C”（十六进制43）而不是NAK来指定CRC-16。
两个字节CRC代替一个字节的算术校验和。

YAM的c选项在单个文件接收时启用CRC-16，对应于MODEM7系列程序中的原始实现。这仍然是默认设置，因为许多商业通信程序和公告板系统仍然不支持CRC-16，尤其是那些用Basic或Pascal编写的。

使用CRC校验的XMODEM协议在发送方和接收方都报告成功传输的情况下是准确的。该协议在分时系统缓冲区过载导致字符丢失的情况下具有鲁棒性。

接收程序生成的单个字符ACK/NAK响应很好地适应了分速调制解调器，其中反向通道的速度限制为主通道速度的百分之十或更少。

XMODEM和YMODEM是半双工协议，它们不会同时尝试在两个方向上传输信息和控制信号。这避免了用户尝试利用全双工异步文件传输协议（如Blast）时出现的缓冲区溢出问题。

Professional-YAM为XMODEM的错误检测和恢复添加了几个专有逻辑增强功能。这些兼容的增强功能消除了在其他程序使用XMODEM协议在非理想条件下进行文件传输时产生的多数不良传输。

4.3 XMODEM-1k1024 字节块

使用YMODEM1从Unix下载时，令人失望的吞吐量导致了1382年1024字节块的开发。102 4字节块通过降低分时系统、调制解调器和分组交换网络对吞吐量的影响，将影响降低了87.5%，同时还将XMODEM在长文件中每字节的开销降低了3%。

使用1024字节块的选择通过发送程序的命令行或选择菜单来表示。2个1024字节块可以提高许多应用程序的吞吐量，但某些环境无法接受1024字节突发，尤其是运行13.2kb端口的迷你计算机。

在传输块的开始处，STX（02）替换了SOH（01），以通知接收方更长的块长度。传输块包含1024字节的数据。接收方应能够接受任何128字节和1024字节块的混合。无论块长度如何，块编号（在块的第二个和第三个字节中）在每个块中增加1。

发送者必须在未收到当前块的合法ACK之前，不得更改128到1024字节块长度。未能遵守此限制会导致传输错误未被检测到。

如果正在使用1024字节块，文件可能会“增长”到下一个1024字节的倍数。如果分配粒度为1k或更大，这不会浪费磁盘空间。在YMODEM批量传输中，文件名块中可选的文件长度允许接收者丢弃填充，保留确切的文件长度和内容。

1024字节块可以用于批量文件传输或单个文件传输。
CRC-16应该与k选项一起使用，以在电话线上保持数据完整性。如果一个程序希望强制执行此建议，它应该在请求者请求CRC-16而不是校验和时取消传输，并发出一个信息诊断消息。

在任何情况下，发送程序不得使用CRC-16，除非接收方命令使用CRC-16。

---

[1]这个名字还没有被创造出来，但协议是相同的。
[2]见下文“KMD/IMP对YMODEM的例外”。

图1XMODEM-1k块

发送者	接收者
	“s -k foo.bar”
“foo.bar 打开 x.x 分钟”	
	C
STX 01 FE Data[1024] CRC CRC	
	ACK
STX 02 FD Data[1024] CRC CRC	
	ACK
STX 03 FC Data[1020] CPMEOF[24] CRC CRC	
	ACK
EOT	
	ACK

图2 混合1024和128字节块

发送者	接收者
	“s -k foo.bar”
“foo.bar 打开 x.x 分钟”	
	C
STX 01 FE Data[1024] CRC CRC	
	ACK
STX 02 FE Data[1024] CRC CRC	
	ACK
SOH 03 FC Data[128] CRC CRC	
	ACK
SOH 04 FB Data[100] CPMEOF[28] CRC CRC	
	ACK
EOT	
	ACK

5 YMODEM 批量文件传输

YMODEM批量协议是XMODEM/CRC协议的一个扩展，允许通过单个命令传输0或多个文件。（如果请求的文件中没有任何一个可访问，则可能不会发送任何文件。）YMODEM批量协议的设计方法是在分层方式下使用发送和接收XMODEM块的正常程序，类似于分组交换方法。

为什么有必要设计一个新的批量协议，而MODEM7中已经有了现有的协议？1MODEM7使用的批量文件模式不合适，因为它不允许传输完整路径名、文件长度、文件日期或其他属性信息。这种过于限制性的设计，仅仅考虑到CP/M，不可能扩展到当前个人计算领域中的Unix、DOS和面向对象系统等领域。此外，MODEM7的批量文件模式在一定程度上容易受到传输干扰。

与单个文件传输的情况类似，接收方通过发送一个“C”字符（用于CRC-16）来启动批量文件传输。

发送方打开第一个文件，并发送包含以下信息的块编号0。仅路径名（文件名）部分是批量传输所必需的。

为了保持向上兼容性，块0中所有未使用的字节必须设置为空。

路径名 路径名（通常是文件名）以空终止的ASCII字符串形式发送。这是由面向句柄的MS DOS（TM）函数和C库fopen函数使用的文件名格式。以下是一个汇编语言示例：

```
DB 'foo.bar'，0
```

路径名中不包含空格。通常情况下，只传输文件名（不含目录前缀），除非发送方已选择 YAM的f选项来发送完整路径名。源驱动器（A:，B:等）不会发送。

文件名考虑因素：

文件名默认转换为小写，除非发送系统支持大小写文件名。这是为了方便使用（例如Unix）系统中存储大小写文件名的用户。

接收方应兼容文件名的大小写。

在不同操作系统之间传输文件时，文件名必须同时被发送方和接收方操作系统接受。

---

[1] MODEM7批处理协议逐个字符地传输CP/MFCB字节f1...f8和t1...t3。接收器逐个回显这些接收到的字节。

[2]只有块的日期部分在这里进行描述。

如果包含目录，它们由/分隔；例如，“subdir/foo”是可接受的，“subdir\foo”则不可。

长度 文件长度以及后续的每个字段都是可选的。1 长度字段以十进制字符串的形式存储在

块中，计算文件中的数据字节数。文件长度不包括任何用于填充最后一个块的CPMEOF (^ Z)或其他垃圾字符。

如果在传输过程中文件正在增长，长度字段应设置为至少最终预期的文件长度，或者不发送。

接收器存储指定的字符数，丢弃发送方添加的任何填充字符以填满最后一个块。修改日期 修改日期是可选的，可以发送文件名和长度，而不需要发送修改日期。
如果发送了修改日期，修改日期与文件长度之间用一个空格隔开。

修改日期以八进制数字发送，表示文件内容最后更改的时间，以1970年1月1日协调世界时（GM T）为起点，以秒为单位。日期为0表示修改日期未知，应保留为文件接收的日期。

此标准格式被选择以消除不同时区之间传输产生的歧义。

模式如果发送文件模式，文件模式与修改日期之间用一个空格隔开。

文件模式以八进制字符串的形式存储。除非文件来自Unix系统，否则文件模式设置为0。rb(1)检查文件模式中的0x8000位，该位表示Unix类型的常规文件。设置了0x8000位的文件假定是从另一个使用相同文件约定的Unix（或类似）系统发送的。这些文件不会以任何方式进行转换。

序列号如果发送了序列号，序列号与文件模式之间用一个空格分隔。发送程序的序列号以八进制字符串的形式存储。

没有序列号的程序应省略此字段，或将其设置为0。接收者使用此字段是可选的。

其他字段YMODEM被设计为允许添加额外的标题字段，如上所述，而不会与较旧的YMODEM程序产生兼容性问题。如需特殊应用需求的其他字段，请联系OmenTechnology。

块的其他部分被设置为空值。这是为了保持向上兼容性。

如果接收到的文件名块带有CRC或其他错误，则请求重传。在接收到文件名块后，如果写入打开成功，则进行确认。如果文件无法

---

1 字段可能不能跳过。

2 如果，偶然地，这些信息超出了128个字节（在Unix4.2BSD扩展文件名中可能发生），则应按上述描述以1k块的形式发送。

打开用于写入，接收方将使用上述描述的CAN字符取消传输。

接收器随后根据标准XMODEM/CRC协议启动文件内容的传输。

文件内容传输完成后，接收器再次请求下一个路径名。空路径名的传输终止批处理文件传输。

请注意，没有文件传输不一定是错误。这可能是因为请求发送者打开的文件都无法读取。

YMODEM接收器默认请求CRC-16。

包含在源代码文件RZSZ.ZOO中的Unix程序sz(1)和rz(1)应该回答有关YMODEM批处理协议的其他问题。

图3YMODEM批量传输会话

发送者	接收者
	“sbfoo.*<CR>
“sending in batch mode etc.”	
	C（命令：rb）
SOH 00 FF foo.c NUL[123] CRC CRC	
	ACK
	C
SOH 01 FE Data[128] CRC CRC	
	ACK
SOH 03 FC Data[128] CRC CRC	
	ACK
SOH 04 FB Data[100] CPMEOF[28] CRC CRC	
	ACK
EOT	
	NAK
EOT	
	ACK
	c
SOH 00 FF NUL[128] CRC CRC	
	ACK

图4YMODEM批量传输会话-1k块

发送者	接收者
	"某人 foo.*<CR>"
“sending in batch mode etc.”	
	C（命令：rb）
SOH 00 FF foo.c NUL[123] CRC CRC	
	ACK
STX 02 FD Data[1024] CRC CRC	
	ACK
SOH 03 FC Data[128] CRC CRC	
	ACK
SOH 04 FB Data[100] CPMEOF[28] CRC CRC	
	ACK
EOT	
	NAK
EOT	
	ACK C
SOH 00 FF NUL[128] CRC CRC	
	ACK

图5YMODEM文件名块由 sz 传输

```
-rw-r--r-- 6347 Jun 17 1984 20:34                bbcsched.txt
0 0100FF62 62637363 6865642E 74787400          |...bbcsched.txt.|
10 36333437 20333331 34373432 35313320         |6347 3314742513 |
20 31303036 34340000 00000000 00000000         |100644..........|
30 00000000 00000000 00000000 00000000
40 00000000 00000000 00000000 00000000
50 00000000 00000000 00000000 00000000
60 00000000 00000000 00000000 00000000
70 00000000 00000000 00000000 00000000
80 000000CA
```

图6 YMODEM头部信息及特性

Program	Length	Date	Mode	S/N	1k-Blk	YMODEM-g
Unix	rz/sz	yes	yes	yes	no	yes
VMS	rb/sb	yes	no	no	no	yes
Pro-YAM	yes	yes	no	yes	yes	yes
CP/M	YAM	no	no	no	no	yes
KMD/IMP	?	no	no	no	yes	no

5.2 KMD/IMP对YMODEM 的例外

KMD和IMP使用接收方发出的"CK"字符序列来触发使用1024字节块作为指定此选项的替代方案。尽管这个两个字符序列在单进程微处理器和直接通信、分时系统和分组交换网络中工作良好，但它们可以在几个秒内分离连续的字符，这使得这种方法不可靠。

如果操作系统环境不阻止可靠实现，发送程序可能会检测到CK序列。

而不是标准的YMODEM文件长度，KMD和IMP在头部块的最后两个字节中传输CP/M记录计数。

6.YMODEM-g 文件传输

发展中的技术正在使用非常专业的技术以越来越高的速度提供电话线路数据传输。这些高速调制解调器，以及像X.PC这样的会话协议，以大大增加延迟时间为代价，提供高速、几乎无错误的通信。

与人类交互相比，这种延迟时间适中，但它破坏了大多数纠错协议的吞吐量。

YMODEM的g选项在这些情况下已被证明是有效的。g选项由接收方驱动，通过发送G而不是C来启动批量传输。
当发送方识别到G时，它会绕过对每个传输块等待ACK的常规操作，以全速发送后续块，但受XOFF/XON或其他由介质施加的流量控制限制。

发送方期望一个初始G来启动特定文件的传输，并且也期望在每个文件末尾发送的EOT上的确认。这种同步允许接收方根据需要打开和关闭文件。

如果在YMODEM-g传输中检测到错误，接收方将使用多个CAN中止序列来中止传输。在需要同时具有流吞吐量和错误恢复的应用程序中应使用ZMODEM协议。

图7YMODEM-g 传输会话

发送者	接收者
	某人foo.*<CR>
“sending in batch mode etc.”	
	G（命令：rb -g）
SOH 00 FF foo.c NUL[123] CRC CRC	
	G
SOH 01 FE Data[128] CRC CRC	
STX 02 FD Data[1024] CRC CRC	
SOH 03 FC Data[128] CRC CRC	
SOH 04 FB Data[100] CPMEOF[28] CRC CRC	
EOT	
	ACK
	G
SOH 00 FF NUL[128] CRC CRC	

7. XMODEM协议概述

1382年8月3日，由沃德·克里斯滕森。

我将保留这份文档的主副本。请通过CBBS/芝加哥(312) 545-8086、CBBS/CPMUG (312) 849-1132或电话(312)849-6279提出更改或建议。

7.1 定义

<soh>	01H
<eot>	04H
<ack>	06H
<nak>	15H
<can>	18H
<C>	43H

7.2传输介质层协议

异步，8位数据，无奇偶校验，一个停止位。

该协议不对传输数据的内容进行限制。在128字节数据消息中不查找控制字符。可以发送任何类型的数据-二进制、ASCII等。该协议尚未正式采用7位环境来传输仅ASCII（或未打包的十六进制）数据，尽管可以通过两端同意在验证之前将协议相关的数据与7F十六进制进行AND操作来实现。我特别指的是校验和以及块号及其反码。

那些希望保持CP/M文件结构兼容性的人，即允许通过调制解调器将ASCII文件从CP/M系统发送或接收，应遵循此数据格式：

使用ASCII制表符（09H）；每8个字符设置一个制表符。

以CR/LF（0DH 0AH）终止的行

以^Z,1AH指示文件结束（一个或多个）

数据是可变长度的，即应被视为一个连续的数据字节流，纯粹为了传输目的被分割成128字节的块。

一个CP/M的“特性”：如果数据正好在128字节边界结束，即CR在127，LF在128，那么包含^Z EOF字符的后续扇区是可选的，但更推荐。一些实用程序或用户程序仍然无法处理没有^Zs的EOF。

最后发送的块与其他块没有区别，即没有“短块”。

图8 XMODEM消息块级协议

传输的每个块看起来像：

<SOH><blk#><255-blk #><--128databytes--><cksum>

其中：

<soh> ＝01 十六进制

<blk #> ＝二进制数，从01开始，每次递增1，并在0FFH处回绕至00H（而不是01）

<255-blk #> ＝经过8080"CMA"指令后的块编号，即每个8位块编号取反。形式上，这被称为“反码”。

<cksum> ＝仅计算数据字节的总和。丢弃任何进位。

7.3文件级协议

7.3.1 发送者和接收者通用

所有错误都会重试10次。对于由操作员运行（即不是XMODEM模式）的版本，在10次错误后，会提示操作员是否要“重试或退出”。

一些协议版本使用<can>，ASCII^X，来取消传输。这从未被采纳为标准，因为只有一个“中止”字符会使传输容易因<ack><nak>或<soh>被损坏成<can>而错误终止传输。

该协议可能被认为是“接收方驱动”，也就是说，发送方不需要自动重传，尽管在当前实现中确实如此。

7.3.2接收程序考虑事项

接收器有一个10秒的超时时间。每次超时都会发送一个nak。接收器的第一次超时，即发送nak，会向发送器发出开始信号。可选地，接收器可以立即发送nak，以防发送器已经准备好。这将节省最初的10秒超时时间。
然而，如果发送器没有准备好，接收器必须继续每10秒超时一次。
一旦接收到一个数据块，接收器将为每个字符和校验和进入一秒的超时。如果接收器因为任何原因（无效的头部、超时接收数据）想要拒绝（nak）一个数据块，它必须等待线路清除。请参阅“编程技巧”以获取想法

同步：如果接收到一个有效的数据块号，它将是：1）预期的那个，在这种情况下一切正常；或者2）之前接收到的数据块的重复。这应该被认为是正常的，仅表明接收器的确认（ack）出现了故障，发送者重新传输；3）任何其他情况块号指示同步丢失的严重情况，例如发送方极少数情况下收到一个

看起来像<ack>的线路故障。中止传输，发送一个<can>。

7.3.3发送程序考虑因素

在等待传输开始时，发送方只有一个非常长的超时时间，比如说一分钟。在当前协议中，发送方在重试前有一个10秒的超时时间。我建议不要这样做，而是让协议完全由接收方驱动。这将兼容现有的程序。

当发送者没有更多数据时，它会发送一个<eot>，并等待一个<ack>，如果没有收到，则会重发<eot>。再次强调，该协议可以是接收者驱动的，发送者只有1分钟的超时时间来终止。

以下是一个数据流的示例，发送一个3块消息。它包括两个最常见的行击中 - 一个垃圾块和一个被垃圾化的<ack>回复。<xx>代表校验字节。

图3 包含错误恢复的数据流

SENDER		RECEIVER
		times out after 10 seconds,
	<---	<nak>
<soh> 01 FE -data- <xx>	--->	
	<---	ack
<soh> 02 FD -data- xx	--->	(data gets line hit)
	<---	<nak>
<soh> 02 FD -data- xx	--->	
	<---	<ack>
<soh> 03 FC -data- xx	--->	
(ack gets garbaged)	<---	<ack>
<soh> 03 FC -data- xx	--->	
	<---	<ack>
<eot>	--->	
	<---	<anything except ack>
<eot>	--->	
	<---	<ack>
(finished)		

7.4 编程技巧

字符接收子程序应使用指定等待秒数的参数调用。接收方应首先使用10秒的时间调用它，然后尝试nak，最多尝试10次。

在接收到<soh>后，接收方应使用1秒的超时时间调用字符接收子程序，用于剩余的消息和<cksum>。由于它们是以连续流的形式发送的，超时意味着出现了严重的类似故障，例如，看到了127个字符而不是128个。

当接收方希望<nak>时，它应该调用一个“清除”子程序，等待行被清除。回想一下，发送方在完成发送一个块后立即将其UART缓冲区中的任何字符丢弃，以确保没有错误被误解。

最常见的技术是让“清除”调用字符接收子程序，指定1秒超时，然后循环回到清除，直到发生超时。然后发送<nak>，确保另一端能看到它。

您可能希望将JohnMahr推荐的一段代码添加到您的字符接收例程中——当UART显示帧错误或溢出时设置一个错误标志。这将有助于捕捉一些更多的故障——其中最常见的是两个连续字节的高位字节发生碰撞。由于以1字节计数产生与将80H+80H相加相同的结果，所以<cksum>的结果是正确的。

[1]这些时间应调整以用于分时系统。

8. XMODEM/CRC 概述

原文 1/13/85 由JohnByrns 创建－CRC选项。

请将此文档中的错误报告或改进建议通过Ward's/CBBS转达给我，电话号码为(312) 849-1132，或通过语音电话(312)885-1105。

在调制解调器协议中使用的CRC是一种块校验的替代形式，它比原始校验和提供了更强大的错误检测能力。安德鲁·S·坦南鲍姆在他的《计算机网络》一书中提到，调制解调器协议使用的CRC-CCITT可以检测所有单比特和双比特错误，所有奇数比特的错误，所有长度为16位或更短的突发错误，99.997%的17位错误突发，以及99.998%的18位和更长的突发错误。

将校验和替换为CRC对调制解调器协议的更改是直接的。
如果这就是我们所做的全部，我们就无法在采用旧校验和协议的程序和采用新CRC协议的程序之间进行通信。为了解决这个问题，增加了一个初始握手。握手允许具有CRC功能的接收程序确定发送程序是否支持CRC选项，并在支持的情况下将其切换到CRC模式。

这个握手设计得可以与仅实现原始协议的程序正确工作。关于这个握手的描述在第10节中给出。

图10 消息块级协议，CRC模式

在CRC模式下，每个传输块看起来像：

<SOH><区块编号><255-区块编号><--128数据字节--><CRC高><CRC低>

其中：

<SOH> = 01十六进制

区块编号 ＝二进制数，从01开始，每次递增1，

循环从0FFH到00H（而不是01）

<255-区块编号> =blk #的补码。

< CRC高> = 包含CRC8高位系数的字节。

< CRC低> = 包含CRC的8个低序系数的字节。

---

1 这个可靠性数据具有误导性，因为XMODEM的关键监管功能并未受到此CRC的保护。

8.1CRC计算

8.1.1正式定义

为了计算16位CRC，消息位被视为一个多项式的系数。
首先将消息多项式乘以X^16，然后使用模二算术除以生成多项式（X^16+X^12+X^5+1）。除法后的余数就是所需的CRC。由于调制解调器协议中的消息块是128字节或1024位，因此消息多项式的阶数为X^1023。消息块第一个字节的最高位是消息多项式中X^1023的系数。消息块最后一个字节的最低位是消息多项式中X^0的系数。

图11 C语言编写的CRC计算示例

以下XMODEMcrc例程取自"rbsb.c"。请参考这些程序的源代码（包含在RZSZ.ZOO中）以了解使用方法。此文件还包括一个快速表驱动的版本。

```
/* update CRC */
unsigned short
updcrc(c, crc)
register c;
register unsigned crc;
{
    register count;
    for (count=8; --count>=0;) {
            if (crc & 0x8000) {
                    crc <<= 1;
                    crc += (((c<<=1) & 0400) != 0);
                    crc ^= 0x1021;
            }
            else {
            crc <<= 1;
            crc += (((c<<=1) & 0400) != 0);
            }
    }
    return crc;
}
```

8.2 CRC文件级协议变更

8.2.1对发送者和接收者都适用

文件级协议中CRC选项的唯一更改是初始握手，用于确定发送和接收程序是否都支持CRC模式。所有调制解调程序都应该支持校验和模式，以与旧版本兼容。希望以CRC模式接收的接收程序通过发送一个<C>代替初始的<nak>来实施模式设置握手。如果发送程序支持CRC模式，它将识别<C>并将其自身设置为CRC模式，并通过发送第一个块作为已收到<nak>来响应。如果发送程序不支持CRC模式，它将不会对<C>做出任何响应。接收程序发送<C>后，它将等待最多3秒钟以接收启动第一个块的<soh>。如果在3秒内收到<soh>，它将假设发送程序支持CRC模式，并将继续以CRC模式进行文件交换。如果在3秒内没有收到<soh>，接收程序将切换到校验和模式，发送一个<nak>，并在校验和模式下继续。如果接收程序希望使用校验和模式，它应该发送一个初始的<nak>，发送程序应该按照原始调制解调协议定义的方式响应<nak>。

在通过初始的<C>或<nak>设置模式后，协议遵循原始ModemProtocol，无论使用校验和还是CRC都是相同的。

8.2.2接收程序考虑事项

模式设置握手至少有4种可能出错的情况。

1. 初始的 <C> 可能会损坏或丢失。
2. 初始的<soh>可能会出现乱码。
3. 初始的 <C> 可以更改为 <nak>。
4. 接收器想要接收校验和的初始nak可以更改为一个C。

如果接收器在第一次超时后发送第二个C，第一个问题就可以解决。
这个过程可以重复多次。但在发送nak并切换到校验和模式或发送没有CRC支持的程序之前，不应重复太多次。如果发送程序通过响应额外的C，就像收到了nak一样，而不是忽略它，重复C也会解决第二个问题。

有可能修复问题3和4，但可能不值得麻烦，因为它们发生的频率非常低。它们可以通过在发送或接收程序中在大量连续nak之后切换模式来修复。然而，这种解决方案可能会带来其他问题。

8.2.3 发送程序注意事项

发送程序应从校验和模式开始。这将确保与仅使用校验和的接收程序的兼容性。在接收到第一个<C>之前，如果接收到<nak>或<ack>，发送程序应将自己设置为CRC模式，并响应如同接收到<nak>一样。
发送方应将额外的 <C> 响应视为 <nak>，直到收到第一个 <ack>。这将帮助接收程序在 <soh>丢失或损坏时确定正确的模式。在收到第一个 <ack> 后，发送程序应忽略 <C>。

8.3带有CRC选项的数据流示例

这里是一个数据流示例，其中接收方请求在CRC模式下进行传输，但发送方不支持CRC选项。此示例还包括各种传输错误。<xx>代表校验字节。

图12 数据流：接收方有CRC选项，发送方没有

发送方		接收方
	<---	<C>
timers out after 3 seconds		
	<---	<C>
timers out after 3 seconds		
	<---	<C>
timers out after 3 seconds		
	<---	<C>
timers out after 3 seconds		
	<---	<nak>
<soh> 01 FE -data- <xx>	--->	
	<---	<ack>
<soh> 02 FD -data- <xx>	--->	(接收到的数据包含错误)
	<---	<nak>
<soh> 02 FD -data- <xx>	--->	
	<---	<ack>
<soh> 03 FC -data- <xx>	--->	
<ack被吃掉了)	<---	<ack>
		10秒超时
	<---	<nak>
<soh> 03 FC -data- <xx>	--->	
	<---	<ack>
<eot>	--->	
	<---	<ack>

这里是一个接收方请求在CRC模式下传输且发送方支持CRC选项的情况下的数据流示例。此示例还包括各种传输错误。<xxxx>代表2个CRC字节。

图13 接收方和发送方都支持CRC选项

发送方		接收方
	<---	<C>
<soh> 01 FE -data- <xxxxx>	--->	
	<---	<ack>
<soh> 02 FD -data- <xxxxx>	--->	(接收到的数据包含错误)
	<---	<nak>
<soh> 02 FD -data- <xxxxx>	--->	
	<---	<ack>
<soh> 03 FC -data- <xxxxx>	--->	
<ack被吃掉了)	<---	<ack>
		10秒超时
	<---	<nak>
<soh> 03 FC -data- <xx>	--->	
	<---	<ack>
<eot>	--->	
	<---	<ack>

9 更多信息

请联系OmenTechnology获取此文档的troff源文件和排版副本。

9.1 TeleGodzilla公告板

更多信息可以通过拨打TeleGodzilla的电话503-621-3746获得。300、1200和2400 bps的速度检测是自动的。TrailBlazer调制解调器用户登录后可以发出TeleGodzilla trailblazer命令切换到13200bps。

有趣的文件包括RZSZ.ZOO、YZMODEM.ZOO、ZCOMMEXE.ARC、ZCOMMDOC.ARC和ZCOMMHLP.ARC。

9.2 UnixUUCP访问

UUCP站点可以通过以下方式获取该文件的当前版本

uucp命令！/u/caf/public/ymodem.doc/tmp

可供使用的文件列表持续更新，存储在/usr/spool/uucppublic/FILES中。当使用uucp检索这些文件时，请记住您的系统目标目录必须可以被任何人写入，否则UUCP传输将失败。

以下L.sys行调用TeleGodzilla（在主机操作中的Pro-YAM）。TeleGodzilla会自动确定传入的速度。

响应“请输入名称:”时，uucico将Pro-YAM的“链接”命令作为用户名给出。

密码（Giznoid）控制着连接到IBMPC其他串行端口的Xenix系统的访问。Pro-YAM与Xenix之间的通信使用3600bps；YAM将其转换为呼叫者的速度。

最后，呼叫的uucico以uucp的身份登录。

Omen任何ACU 24001-503-621-3746 se:--se:链接ord:Giznoid in:--in:uucp

10. 修订

1987年8月3日 修订以增强清晰度。

1987年5月31日 强调了YMODEM的最小要求，并更新了访问文件的信息。

1986年3月11日明确了命名法以及一些小问题。

1986年4月15日的版本明确了有关CRC计算和标题中空格的一些问题。

11. YMODEM程序

ZCOMM，一个类似于Professional-YAM的共享软件小兄弟，可在TeleGodzilla和其他公告板系统上以ZCOMMEXE.ARC的形式获得。ZCOMM可用于测试YMODEM和ZMODEM的实现。

Unix支持YMODEM的程序可在TeleGodzilla的RZSZ.ZOO上找到。这个ZOO存档包括一个 ZCOMM/Pro-YAM/PowerCom脚本ZUPL.T，用于上传MINIRB.C引导程序，编译它，然后使用编译后的MINIRB上传其余文件。大多数类似Unix的系统都得到支持，包括V7、Xenix
、SysIII、4.2BSD、SYSV、Idris、Coherent和Regulus。

在VRBSB.SHQ中有一个VAX-VMS版本。

艾尔文·霍夫为KMD和IMPseries程序添加了1k块和基本的YMODEM批量传输功能，分别替代了XMODEM和MODEM7/MDM7xx系列。
提供了适用于各种CP/M系统的覆盖文件。

关于Professional-YAM通讯软件的问题可以联系：

查克·福斯伯格
奥门科技有限公司
17505-VSauvieIslandRoad
波特兰 俄勒冈州37231
电话：503-621-3406 ：
语音调制解器：503-621-3746 速度：
2400,1200,300
Usenet：...!tektronix!reed!omen!c af
CompuServe:70007,2304
GEnie:CAF

与ZMODEM和Kermit不同，XMODEM和YMODEM给可靠的高性能实现设置了障碍，这在行业领导者XMODEM和YMODEM程序在压力下的可靠性低下得到了证明。OmenTechnol ogy为希望以最先进的功能和可靠性实现XMODEM、YMODEM和ZMODEM的人提供咨询服务和其他服务。

中文译者 miuser

www.miuser.net

文中翻译错误请联系64034373@qq.com修正