SylixOS 简要说明

实时系统与分时系统的区别

实时系统（Real-time operating system,RTOS）的正确性不仅依赖系统计算的逻辑结果，还依赖于产生这个结果的时间。换句话说，系统设计时所有的事件都可以在指定的时间内得到响应。如果系统关键任务响应时间都满足这条标准，则这样的实时系统可称为硬实时系统。

与通用的分时操作系统不同（Linux、Windows、Unix 等），实时操作系统在航空航天、军事与工业自动化领域更具优势，首先实时操作系统有着分时操作系统无法比拟的响应时间确定性，实时操作系统从调度器算法，到中断响应系统，到消息传递机制等所有的核心算法时间复杂度都是 O（1），它表示系统的响应速度不依赖于系统任务的多少，负载的轻重，而只依赖于优先级的设计，就算当前系统满负荷运行，优先级高的事件发生后，系统还将会在指定的时间内立即响应事件。由于这种设计理念和算法上的优势，根据相关数学理论，分时系统在负载严重的情况下是不能通过提升处理器性能来获得确定的响应时间。

这种算法上的优势是通用分时系统所难以比拟的，而分时系统则更多考虑的是系统易用性、平衡性和数据吞吐率。所以实时系统与分时系统设计思想和应用领域完全不同，不存在替代关系，而是一种互补关系。

SylixOS 简介

SylixOS 是一款为嵌入式系统设计的硬实时操作系统（RTOS）。此系统于 2006 年开始开发工作。设计之初只是为了验证相关操作系统算法，后来经过多年的持续开发与改进，SylixOS 已经不只是一个实时操作系统，它已经成为一个可靠稳定，功能全面，易于开发调试的实时嵌入式系统开发平台（SylixOS 至今依然保持以开放源代码的形式存在）。

SylixOS 的诞生可以摆脱国内一些关键性设备对国外嵌入式操作系统的依赖，为国内的嵌入式信息技术行业提供一个全新的选择。

目前 SylixOS 已经成功应用于工业控制与通信、武器装备及国家安全、新能源应用等国家基础领域中。

需要说明的是：为了保证 SylixOS 能够持续开发，并且吸引大批开发人员参与测试，所以 SylixOS 目前是以公开源代码项目的形式存在。

SylixOS 定位

SylixOS 是一款嵌入式硬实时操作系统，同其类似的操作系统，全球比较知名的还有 VxWorks(主要应用于航空航天、军事与工业自动化领域)、RTEMS(起源于美国国防部导弹与火箭控制实时系统)、ThreadX(主要应用于航空航天与数码通讯)等。

从全球范围上看，SylixOS 作为实时操作系统的后来者，在设计思路上借鉴了众多实时操作系统的设计思想，其中就包括 RTEMS、VxWorks、ThreadX 等，使得具体性能参数上达到或超过了众多实时操作系统的水平，成为国内实时操作系统的最优秀代表之一。

当前主要功能与特点

SylixOS 作为抢占式多任务硬实时操作系统，具有如下功能与特点：

1. 兼容 IEEE 1003（ISO/IEC 9945）操作系统接口规范
2. 兼容 POSIX 1003.1b（ISO/IEC 9945-1）实时编程的标准
3. 优秀的实时性能（任务调度与切换、中断响应算法都是 O（1）时间复杂度算法）
4. 支持无限多任务
5. 抢占式调度支持 256 个优先级
6. 支持协程（windows 称为纤程）
7. 支持虚拟进程
8. 支持优先级继承，防止优先级翻转
9. 极其稳定的内核，很多基于 SylixOS 开发的产品都需要 7x24 小时不间断运行
10. 内核 CPU 占用率低
11. 柔性体系(Scalable)
12. 核心代码使用 C 编写，可移植性好
13. 支持紧耦合同构多处理器（SMP），例如：ARM Cortex-A9 SMPCore
14. 全世界独一无二的硬实时多核调度算法
15. 支持标准 I/O、多路 I/O 复用与异步 I/O 接口
16. 支持多种新兴异步事件同步化接口，例如：signalfd、timerfd、eventfd 等
17. 支持众多标准文件系统：FAT、YAFFS、ROOTFS、PROCFS、NFS、ROMFS 等等
18. 支持文件记录锁,可支持数据库
19. 支持统一的块设备 CACHE 模型
20. 支持内存管理单元（MMU）
21. 支持第三方 GUI 图形库，如：Qt、Microwindows、μC/GUI 等等
22. 支持动态装载应用程序、动态链接库以及模块
23. 支持扩展系统符号接口
24. 支持标准 TCP/IPv4/IPv6 双网络协议栈，提供标准的 socket 操作接口
25. 支持 AF_UNIX, AF_PACKET, AF_INET, AF_INET6 协议域
26. 内部集成众多网络工具，例如：FTP、TFTP、NAT、PING、TELNET、NFS 等等
27. 内部集成 shell 接口、支持环境变量（与 Linux 操作习惯基本兼容）
28. 内部集成可重入 ISO/ANSI C 库（支持 80%以上标准函数）
29. 支持众多标准设备抽象，如：TTY、BLOCK、DMA、ATA、GRAPH、RTC、PIPE 等。同时支持多种工业设备或总线模型，如：CAN、I2C、SPI、SDIO 等
30. 提供高速定时器设备接口，可提供高于主时钟频率的定时服务
31. 支持热插拔设备
32. 支持设备功耗管理
33. 内核、驱动、应用程序支持 GDB 调试
34. 提供内核行为跟踪器，方便进行应用性能与故障分析

适合应用领域

SylixOS 采用抢占式多任务硬实时的方式来设计整个操作系统。其技术实现的核心目标

是实时可控，稳定可靠。所以 SylixOS 适用于以下对实时性与稳定性的要求尤为突出的领域：

1. **工业实时控制领域**：主要包括工业机器人系统、现场安全监控与防护系统、工业现场总线通信管理系统等；
2. **航空航天领域**：主要包括航空器姿态控制系统、航空航天数据记录仪系统、高精度测绘系统，航空航天通信系统等；
3. **国家安全领域**：主要包括加密通信系统、无线传感器终端分析系统、虚拟仪表系统、军事数据记录系统、火控系统等；
4. **金融收费领域**：主要包括 POS 收费系统、终端支付系统等；
5. **高端民用领域**：主要包括汽车行驶记录仪系统、车辆及船用发动机中央 ECU 系统（OSEK 标准）、生产线测试系统、分布式无人值守系统等；

当前应用实例

从 2006 年到现在已经有很多产品基于 SylixOS 进行开发，涵盖领域广泛，产品稳定可靠，下面将从工业自动化、军事、通信、民用等领域介绍基于 SylixOS 的小部分产品。其中大部分产品都要求 7x24 小时不间断运行，当前很多 SylixOS 系统节点甚至不间断运行已超过 40000 多小时（5 年多）。

工业自动化领域：

1. 通用组态开发人机界面
2. ABB 定制火灾报警系统 ACDU
3. 计重收费与超限检测仪
4. 特种车辆与船用发动机状态显示器
5. 动力环境监控站
6. 门禁系统事件服务器
7. 通用 PLC 系统内核

……

多媒体领域：

1. 电梯广告机
2. 矿井网络可视通话系统

……

新能源领域

1. 小型光伏发电实时数据管理器（日本、欧盟）
2. 大型光伏发电节点管理器（北美）

3. 世博会申沃超级电容车监控系统

……

现场通信系统

1. 多种工业现场总线协议转换器
2. 煤矿井下无线人员定位系统

……

与现有同类操作系统对比

当前国外存在多种嵌入式实时操作系统，每种系统都有各自的优缺点，以下的对比表将通过一些关键参数对比几个典型的实时操作系统：

	SylixOS	VxWorks	RTEMS	µC/OS-II	eCos
内核抢占	■	■	■	■	■
优先级	256	256	256	64	32
优先级变化	动态	动态	动态	动态	动态
优先级继承	■	■	■	天花板算法	■
同优先级	■	■	■	□	■
任务数量	无限	无限	无限	64	无限
进程支持	POSIX 进程	RTP 进程	□	□	□
协程（纤程）	■	□	□	□	□
MMU 管理	■	■	□	□	□
内存映射	■	■	□	□	□
SMP 多核	■实时调度	■	协作式多核	□	□
SMP-RT 调度	■	暂不确定	□	□	□
RMS 调度	■	□	■	□	□
时间确定性	■	■	■	■	■
I/O 系统	■ 95%以上兼容 POSIX 标准与 扩展接口	■ 70%左右兼容 POSIX 标准	■ 70%左右兼容 POSIX 标准	□	不完全支持
同步	计数信号量 二值信号量 互斥信号量 事件标志 eventfd	计数信号量 二值信号量 互斥信号量 事件标志	计数信号量 二值信号量 互斥信号量 事件标志	计数信号量 事件标志	计数信号量 二值信号量 事件标志
通信量	消息队列 管道 流式管道 socket posix mqueue	消息队列 管道 socket posix mqueue	消息队列 管道 socket posix mqueue	消息队列	邮箱

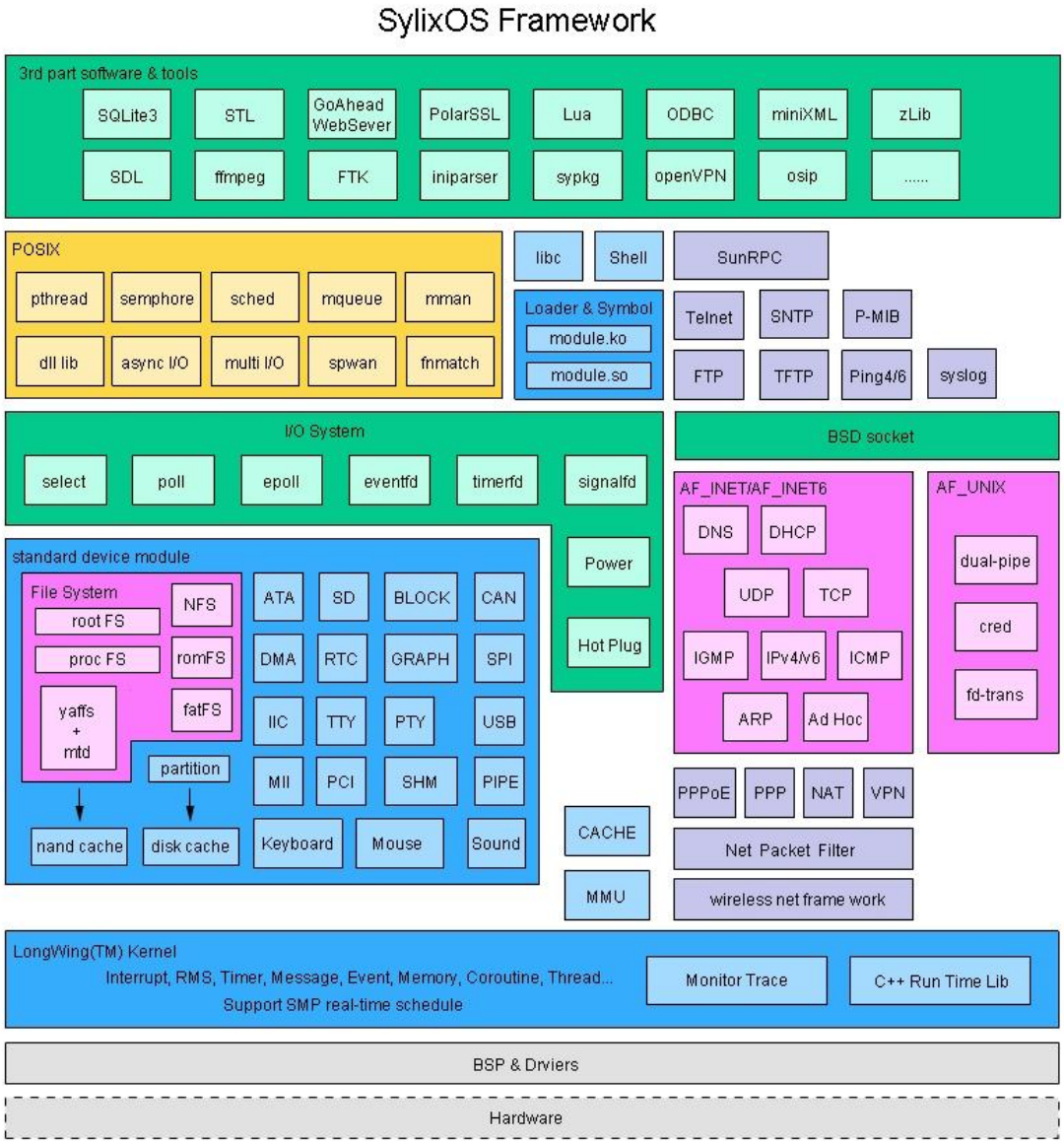
多文件系统	■	■	■	□	不全面
信号系统	健全	健全	不健全	无	无
异步 I/O	■	■	□	□	□
高速定时器	■	□	□	□	□
ProcFS	■	■	□	□	□
写平衡 FS	Yaffs	Tffs	□	□	□
第三方支持	多	较多	较多	少	少
实时数据库	■	■	■	□	□
ODBC	■	□	□	□	□
动态装载	■	■	□	□	□
动态链接库	■	■	□	□	□
UNIX 域协议	■	■	□	□	□
AF_PACKET	■	□	□	□	□
描述符传递	■	□	□	□	□
TCP/IP	■	■	■	□	■
IPv6	■	■	□	□	□
POSIX	■	■	■	□	部分
网络工具集	■	■	■	□	■
shell	■	■	■	□	□
libc	■	■	第三方	第三方	第三方
USB	■	■	□	□	□
PCI	■	■	■	□	■
SD-BUS	■	□	□	□	□
CAN	■	■	□	□	□
多种块设备	■	■	□	□	□
工业总线	CAN 以太网	CAN 以太网	以太网	□	□
Mesh 网络	■	■	□	□	□
自组网协议	MAODV	□	□	□	□
规则滤波器	■	■	■	□	□
内置热插拔	■	□	□	□	□
文件记录锁	■	不全面	□	□	□
I/O 多路复用	■	■	■	□	□
C++支持	■	■	■	□	□
多媒体	支持对应设备与驱动标准	支持对应设备驱动	□	□	□
Unix 程序兼容性	较好	较好	一般	不支持	较差
编译器	GCC 等	专用编译器	GCC	C 编译器	GCC
调试	GDB	WDB	GDB	□	□
内核跟踪器	■	■	□	□	□
脚本语言支持能力					

Lua	■	不完整	不完整	□	不完整
Python	■	□	□	□	□
Tcl	未验证	□	□	□	□

软件框架介绍

SylixOS 使用微内核设计，内核简洁高效，运行稳定快速，操作系统本身支持编译时裁剪。同时由于 SylixOS 是实时操作系统中少有的支持进程与动态装载的操作系统，所以整个系统支持运行时可裁剪。

SylixOS 系统框架图如下图所示。



如上图所示，SylixOS 内核（longwing）小巧，本身只提供基本的操作系统服务，例如：线程管理，基本的线程间通信，事件管理，中断管理，内存管理，多核实时调度器等，同时 SylixOS 内核包含一个基础的 C++运行时库，所以 SylixOS 内核模块支持不带有异常处理和运行时类型识别功能的 C++程序。

内核上层所有服务均为可裁剪服务，例如 I/O 系统，标准设备，网络协议栈，POSIX 兼

容层等等。

接下来的章节会详细介绍 SylixOS 各个功能组建的功能与特点。

POSIX 标准

在介绍 SylixOS 各个功能组件前，这里必须首先说明什么是 POSIX 系统。POSIX 是 IEEE 为了要在各种 UNIX 操作系统上运行的软件，而定义 API 的一系列互相关联的标准的总称，其正式称呼为 IEEE 1003，而国际标准名称为 ISO/IEC 9945。此标准源于一个大约开始于 1985 年的项目。POSIX 这个名称是由理查德·斯托曼应 IEEE 的要求而提议的一个易于记忆的名称。它基本上是 Portable Operating System Interface（可移植操作系统接口）的缩写，而 X 则表明其对 Unix API 的传承。

其中 POSIX 对实时操作系统有一个子协议称作 1003.1b，它定义了标准实时操作系统的基本行为，SylixOS 符合此协议要求。

当前的 POSIX 主要分为四个部分：Base Definitions、System Interfaces、Shell and Utilities 和 Rationale。SylixOS 兼容这四部分的绝大多数 API。

目前符合 POSIX 的操作系统有：UNIX、BSD、LINUX、iOS、Android、SylixOS、VxWorks、RTEMS 等等，由于 SylixOS 支持 POSIX 所以 SylixOS 的应用在这些操作系统上非常好移植。

POSIX 对操作系统基本行为做出了较为严格的规定，这些规定以 API 形式给出。同时提供了对标准 API 引用的头文件。这些头文件分为四组，分别称作：ISO C 标准头文件、POSIX 必须头文件、POSIX XSI 扩展头文件、POSIX 实时标准头文件。
这些头文件分别是：

ISO C 标准头文件

<assert.h>	验证程序断言（SylixOS 支持）
<complex.h>	支持复数算术运算（libm 支持）
<ctype.h>	字符类型（SylixOS 支持）
<errno.h>	出错码（SylixOS 支持）
<fenv.h>	浮点环境（libm 支持）
<float.h>	浮点常量（gcc 支持）
<inttypes.h>	整型格式转换（SylixOS 支持）
<iso646.h>	替代关系操作符宏（gcc 支持）
<limits.h>	实现常量（SylixOS 支持）
<locale.h>	局部类别（SylixOS 支持）
<math.h>	数学常量（libm 支持）
<setjmp.h>	非局部 goto（SylixOS 支持）
<signal.h>	信号（SylixOS 支持）
<stdarg.h>	可变参数表（SylixOS 支持）
<stdbool.h>	布尔类型和值（gcc 支持）
<stddef.h>	标准定义（gcc 支持）
<stdint.h>	整型（SylixOS 支持）
<stdio.h>	标准 I/O 库（SylixOS 支持）
<stdlib.h>	实用程序库函数（SylixOS 支持）

<string.h> 字符串操作（SylixOS 支持）
<tgmath.h> 通用类型数学宏（libm 支持）
<time.h> 时间和日期（SylixOS 支持）
<wchar.h> 扩展的多字节和宽字符支持（SylixOS 支持）
<wctype.h> 宽字符分类和映射支持（SylixOS 支持）

POSIX 必须头文件

<dirent.h> 目录项（SylixOS 支持）
<fcntl.h> 文件控制（SylixOS 支持）
<fnmatch.h> 文件名匹配类型（SylixOS 支持）
<glob.h> 路径名模式匹配类型（SylixOS 支持）
<grp.h> 组文件（SylixOS 支持）
<netdb.h> 网络数据库操作（SylixOS 支持）
<pwd.h> 口令文件（SylixOS 支持）
<regex.h> 正则表达式（SylixOS 支持）
<tar.h> tar 归档值（SylixOS 支持）
<termios.h> 终端 I/O（SylixOS 支持）
<unistd.h> 符号常量（SylixOS 支持）
<utime.h> 文件时间（SylixOS 支持）
<wordexp.h> 字扩展类型（SylixOS 支持）
<arpa/inet.h> Internet 定义（SylixOS 支持）
<net/if.h> 套接字本地接口（SylixOS 支持）
<netinet/in.h> Internet 地址族（SylixOS 支持）
<netinet/tcp.h> 传输控制协议定义（SylixOS 支持）
<sys/mman.h> 内存管理声明（SylixOS 支持）
<sys/select.h> select 函数（SylixOS 支持）
<sys/socket.h> 套接字接口（SylixOS 支持）
<sys/stat.h> 文件状态（SylixOS 支持）
<sys/times.h> 进程时间（SylixOS 支持）
<sys/types.h> 基本系统数据类型（SylixOS 支持）
<sys/un.h> UNIX 域套接字定义（SylixOS 支持）
<sys/utsname.h> 系统名（SylixOS 支持）
<sys/wait.h> 进程控制（SylixOS 支持）

POSIX XSI 扩展头文件

<cpio.h> cpio 归档值（SylixOS 支持）
<dlfcn.h> 动态链接（SylixOS 支持）
<fmtmsg.h> 消息显示结构（SylixOS 支持）
<ftw.h> 文件树漫游（SylixOS 支持）
<iconv.h> 代码集转换实用程序（SylixOS 支持）
<langinfo.h> 语言信息常量（SylixOS 支持）
<libgen.h> 模式匹配函数定义（SylixOS 支持）

<monetary.h>	货币类型（SylixOS 支持）
<ndbm.h>	数据库操作（SylixOS 支持）
<nl_types.h>	消息类别（SylixOS 支持）
<poll.h>	轮询函数（SylixOS 支持）
<search.h>	搜索表（SylixOS 支持）
<strings.h>	字符串操作（SylixOS 支持）
<syslog.h>	系统出错日志记录（SylixOS 支持）
<ucontext.h>	用户上下文
<ulimit.h>	用户限制（SylixOS 支持）
<utmpx.h>	用户帐户数据库（SylixOS 支持）
<sys/ipc.h>	IPC（libxsiipc.ko 内核模块支持）
<sys/msg.h>	消息队列（libxsiipc.ko 内核模块支持）
<sys/resource.h>	资源操作（SylixOS 支持）
<sys/sem.h>	信号量（libxsiipc.ko 内核模块支持）
<sys/shm.h>	共享存储（libxsiipc.ko 内核模块支持）
<sys/statvfs.h>	文件系统信息（SylixOS 支持）
<sys/time.h>	时间类型（SylixOS 支持）
<sys/timeb.h>	附加的日期和时间定义（SylixOS 支持）
<sys/uio.h>	矢量 I/O 操作（SylixOS 支持）

POSIX 实时标准头文件

<aio.h>	异步 I/O（SylixOS 支持）
<mqueue.h>	消息队列（SylixOS 支持）
<pthread.h>	线程（SylixOS 支持）
<sched.h>	执行调度（SylixOS 支持）
<semaphore.h>	信号量（SylixOS 支持）
<spawn.h>	实时 spawn 接口（SylixOS 支持）
<stropts.h>	XSI STREAMS 接口
<trace.h>	时间跟踪

内核服务

SylixOS 内核小巧，它提供的操作系统最基础的服务，这些服务包括：

1. **线程管理**
2. **协程管理**
3. **事件标志组管理**
4. **中断管理**
5. **调度器**
6. **内存管理**
7. **消息队列**
8. **计数、互斥、二值信号量**
9. **定时器管理**

10. 资源回收器

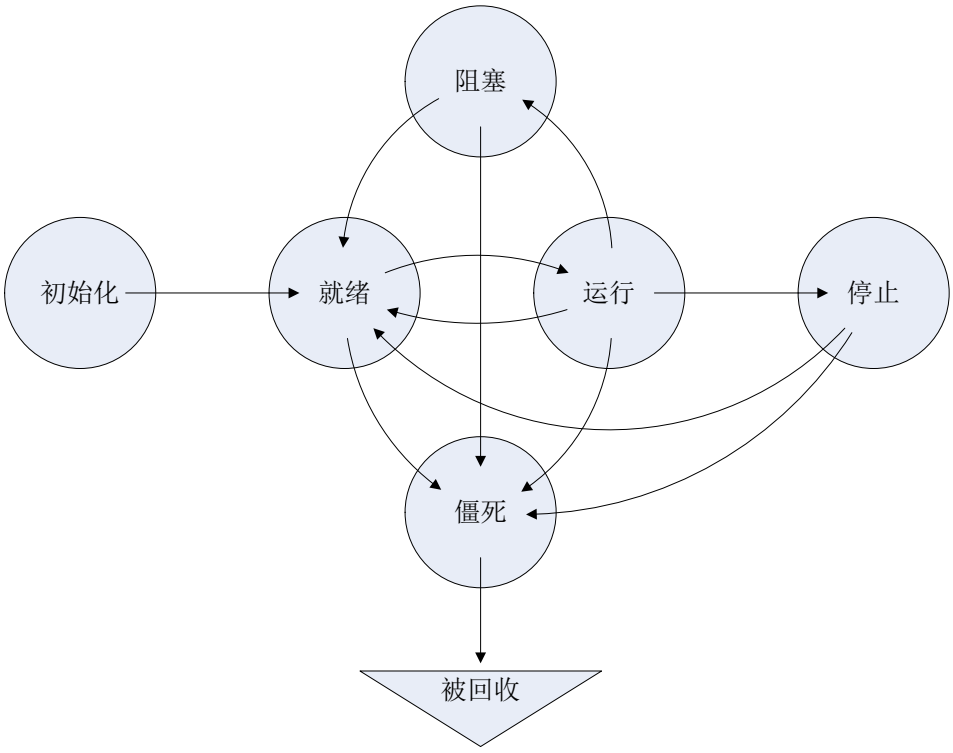
这些最基础的功能构成了 SylixOS 所有功能与服务的核心。SylixOS 本身是一个实时操作系统，所以内核调度器（scheduler）使用基于优先级的抢占式调度算法，调度器调度的基本单元为线程。SylixOS 永远运行优先级最高的线程。

SylixOS 调度器支持紧耦合同构多处理（SMP）并且调度器调度时间复杂度为 O（1），换句话说，调度器每次调度消耗的时间与需要调度的线程总数量没有关系，即调度时间确定，这种系统适合于对时间有严格要求的工业与军事系统。此调度器同样支持同优先级线程，这些线程可按先进先出或者时间片轮转调度算法。

SylixOS 线程（thread）有以下六种状态：

1. **初始化**
2. **就绪**
3. **运行**
4. **阻塞**
5. **僵死**
6. **停止**

线程各状态迁移顺序由下图所示：



【初始化】状态表示一个线程刚刚创建，操作系统为它分配了运行所必须的资源但是没有将它送入调度器。此时用户可以获取线程句柄，选择合适的时间启动线程。很多操作系统的线程没有这种状态。SylixOS 加入这种状态主要是为了适应“静态系统”的要求，所谓静态系统就是运行线程确定，所需资源确定，行为确定的系统，例如汽车电子控制单元（ECU）系统。这类系统对响应和可靠性要求非常高，所以 SylixOS 可以提前分配好线程所有需要的资源，为它的运行做好一切准备，这样就不会因为资源短缺造成严重后果。

【就绪】状态表示一个线程可以被执行，即线程得到了除 CPU 以外运行所需要的所有资源，例如线程等待的信号量已经有效，这时线程请求调度器调度，至于调度器合适将 CPU 资源提供个这个线程以使其运行，则依赖于上面所说的调度器算法。

【运行】状态表示一个线程正在执行，即就绪后，系统调度器为这个线程分配了 CPU 资源使其得以运行。

【阻塞】状态表示线程现在不能继续执行，必须需要等待指定的事件发生，例如线程正在等待信号量有效，或者消息队列中的信息，或者一个信号，一个同步 I/O 请求等等。这个状态下线程得不到执行，除非等待的事件到达或者设置的超时时间到达，如果这两种情况发生，则线程进入就绪队列。等待调度器再次执行，同时将等待的结果告知用户代码。

【僵死】线程结束时需要回收操作系统在创建它时分配的资源，有些资源是它自身运行时无法回收的，例如内核对象，堆栈等，这时线程通知操作系统相关的回收器，并把自己设置为僵死状态等待回收器彻底销毁自己。

【停止】状态表示线程被其他线程或者系统强行暂停执行，此状态只有 SMP 系统会出现，当一个核正在运行的线程（或进程）向另一个核上的线程发送信号时，这时发送信号的目标线程正在另一个核上运行，则内核会自动将目标设置为停止状态，然后等信号发送完毕后再将这个任务恢复执行。

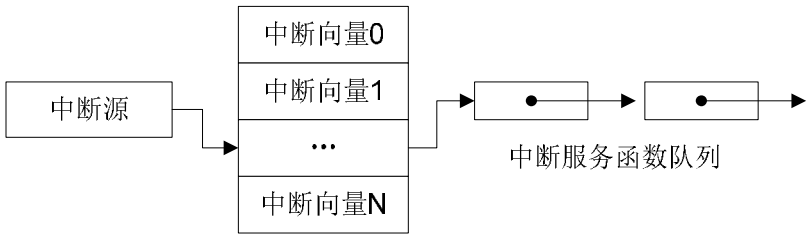
SylixOS 系统上线程实体的状态永远是以上六种状态之一。

协程（coroutine），又称作协同程序是比线程还小的可执行代码序，在 windows 操作系统上称为纤程。一个线程内可以拥有多个协程，这些协程共享线程除了栈之外的所有资源，例如优先级，内核对象等等。由于线程内的所有协程共享线程本身的内核对象，所以调度器本身并不知道协程的存在，协程的运行是靠所属线程被调度时被执行的。一个线程内的协程共享所属线程的优先级，一个线程内部的协程不可被强占。只能轮转运行。而且当前正在运行协程必须主动放弃 CPU 另同线程内的另一个协程才能得以运行，当线程被删除时，线程内的所有协程也同时被全部删除。

SylixOS 在内核中引入协程概念，而不是使用库模拟出的协程，这样 SylixOS 内部的协程管理更加便捷高效。

事件标志组（eventset）是 SylixOS 提供的一个线程同步通信机制。每一个事件标志组包含有 32 位事件标志，在应用中每一位可以代表一个事件，线程可以等待事件标志组中的一个事件或者同时等待多个事件。线程等待多个事件时既可以等待多个事件同时发生，也可以等待多个事件中任何一个事件的发生。指定的事件发生后，等待相应事件的线程将会从阻塞状态被转换成就绪状态。这时只要调度器调度该线程，它就可以执行。

SylixOS 中断系统结构如下图所示，它本身可以支持无限数量的中断（interrupt）源，具体的中断向量表（interrupt vector）大小由编译时的配置决定。



SylixOS 和大多数操作系统一样使用平板中断向量表，如果系统硬件中存在有主从级联中断，则需要在板级支持包（BSP）中进行相关的抽象。当系统发生中断时，BSP 代码按操作系统要求处理好相关中断上下文（context）后，调用操作系统统一的中断入口 API。操作系统根据中断向量号来决定运行哪些之前已经注册的中断服务。驱动程序不需要操作中断的具体行为，只需要注册相关的中断向量即可。

SylixOS 同时支持单向量多中断服务函数表，所以像 PCI 总线这样的多级中断系统，使

用此类中断向量管理，用户驱动程序将变的非常简单。

SylixOS 调度器只管理一种资源：CPU。调度器决定当前时刻将 CPU 分配给哪个线程，即运行哪个线程。每一个 CPU 核心只能同时运行一个线程，如果没有任何就绪的用户线程，则调度器让 CPU 运行永远就绪的空闲（idle）线程。

如上所述，对于单核 CPU 系统，一般的多任务操作系统在宏观上是多任务并行处理，但在微观上，同时只有一个线程运行。调度器根据调度算法和当前就绪线程的相关参数来判断谁将运行。对于紧耦合多 CPU 系统（SMP，例如酷睿，龙芯多核），他们可以同时运行与内核数量相当的线程，例如双核处理器可以同时运行两个线程，如果这两个线程没有资源冲突，则理论运行速度将达到单核两倍（真正的并行处理系统），但是没有资源冲突的线程是理想状态，所以真实运行的 N 核系统是不可能达到单核系统 N 倍的效率。对于多核系统，SylixOS 调度器将决定这些核同时运行哪些线程。对于硬实时的 SylixOS 来说，调度器将 CPU 分配给就绪线程中优先级最高的几个线程，这个调度算法时间复杂度也为 O（1）。

SylixOS 提供三种形式的内存管理：堆内存管理（heap），定长分区内存管理（pool），虚拟内存管理（vmm）。这三种内存管理有着不同的用途。

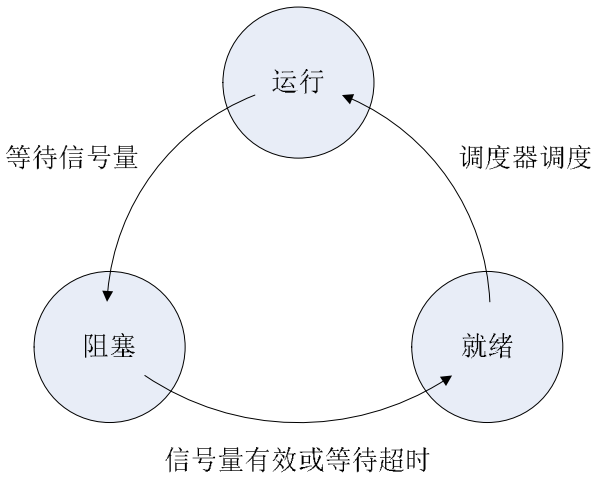
堆内存管理类似于我们常用的 malloc、free 操作，操作系统有两个非常关键的内存堆：“内核内存堆”和“系统内存堆”。内核堆负责系统内核对象的缓冲。系统堆负责内核基础缓冲区，驱动程序缓冲区等。

虚拟内存管理只用于有硬件内存管理单元（MMU）的处理器。它管理了整个虚拟内存空间和物理页面的分配，回收，映射，权限。是操作系统内存管理的核心。例如后面将要提到的进程，文件映射，缺页中断系统等等，都需要虚拟内存管理的支持。同时它也接管了所有内存的异常访问处理（映射错误或者权限错误）。

SylixOS 物理内存采用页式内存管理算法。物理页面使用伙伴算法，虚拟空间采用哈希红黑二叉树保证分配与回收的速度。

消息队列是 SylixOS 提供的多任务同步通信方式之一，它将一个消息从一个任务传递到另一个任务，这样一个线程可以将产生的数据或者事件发送出去，由指定的任务继续处理。这也是操作系统多任务最常用的同步通信方式。

信号量也是 SylixOS 提供的多任务同步通信方式之一，SylixOS 的信号量分为三种：计数信号量、二值信号量、互斥信号量。信号量既可以用作任务间的通信，也可以用于中断与任务的通信。SylixOS 有关信号量对线程状态的影响如下图所示。



计数信号量内部是一个 32/64 位计数器，当计数值为 0 时等待的线程将被阻塞，如果不为 0 则计数器减一线程继续运行。发送信号量时，如果此时有线程阻塞在对应的信号量上，

则阻塞的任务将被激活，如果没有则计数器加一。

二值信号量顾名思义，它只有两种状态：有效和无效。线程等待此信号量时如果信号量有效，则信号量立即变为无效状态，同时当前线程不阻塞并继续执行。如果信号量是无效状态，则线程被阻塞。发送信号量时，如果此时有线程阻塞在对应的信号量上，则阻塞的线程将被激活，如果没有等待此信号量的线程，则将信号量置为有效状态。

互斥信号量（简称互斥量）是 SylixOS 为保护共享资源设计的一种锁机制（大多数操作系统拥有此功能），互斥信号量的基本行为类似于二值信号量，但是功能更加强大。但它并不能取代二值信号量，因为释放互斥信号量的线程必须是之前获取该信号量的线程，所以互斥量不能被用作线程同步通信。SylixOS 的互斥量提供两种方法避免优先级倒置（又名优先级反转），一种是优先级继承算法，一种是优先级天花板算法，创建互斥量时通过选项来选择需要使用的算法。同时互斥量支持资源死锁检测，死锁可重入功能，同样也是根据创建选项确定。

SylixOS 提供两种类型的定时器：高速内核定时器，任务级定时器。高速内核定时器顾名思义，它的定时周期可以非常短，频率可以高过操作系统的时钟。但需要注意的是，高速定时器的用户回调函数可能运行在中断上下文中（由 BSP 代码决定），所以只能进行简单的操作，一般用于发送信号量激活等待线程，或者通知什么事情发生。

任务级定时器速度较慢，他提供普通精度的定时要求。定时器服务函数运行在操作系统 t_timer 内核线程中。

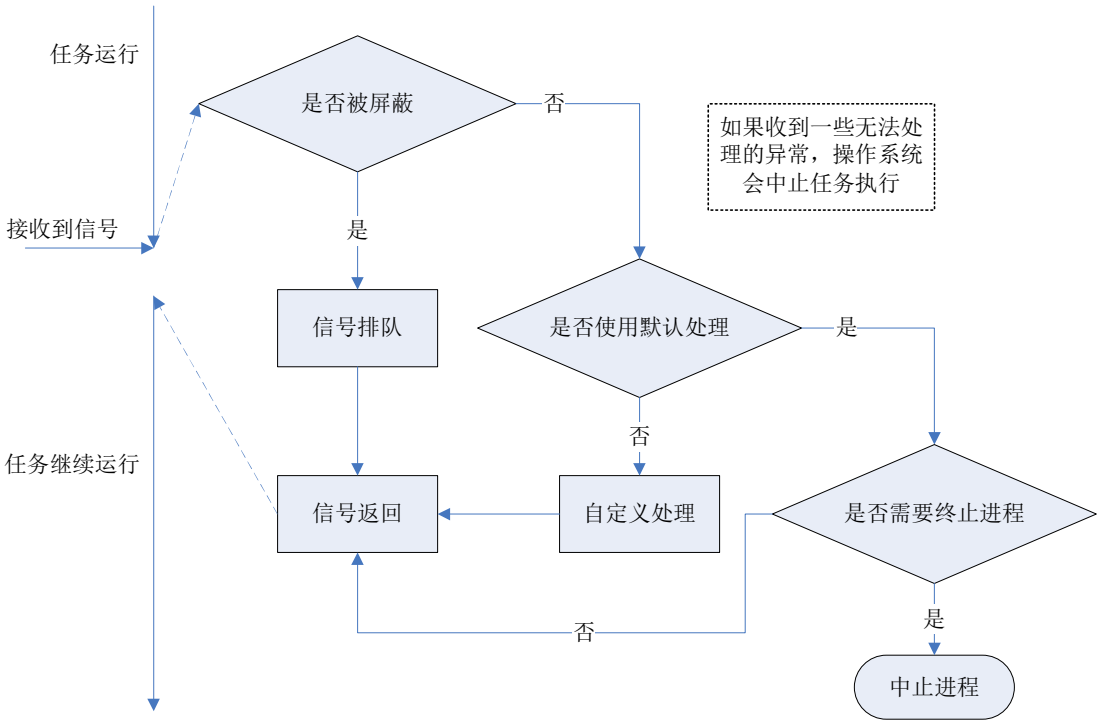
需要注意的是：SylixOS 内核提供的定时器只用于内核模块或者内核应用，进程需要使用与信号系统相关的 posix 定时器。

SylixOS 的回收器主要是记录一个进程的资源使用情况，在进程退出时将释放进程的所有资源，它们包括内核对象，文件描述符，进程空间，依赖的动态链接库等等。进程相关的内容将在下面进程一节详细讲解。

信号系统

信号是一种异步通信方式，也是 POSIX 兼容操作系统唯一的异步通信方式（之后章节只要提到异步通信，例如异步 I/O 都是信号驱动的）。SylixOS 支持 POSIX 标准所定义的信号系统。

信号实际上是一种在软件层次上模拟的中断，其处理流程如下图所示。



所谓异步通信是指：应用程序不用等待事件的发生，当信号发生时应用程序自动陷入到对应的信号处理句柄中，就相当于应用程序被中断了一样。很多重要的系统级应用程序都要处理信号。信号提供了一种处理异步事件的方法。SylixOS 共提供 64 个不同类型的信号，分别对应系统产生的不同类型的异常，例如：定时器到时，异步 I/O 操作完成，管道断裂，内存非法访问，除 0 操作，子进程结束等等。

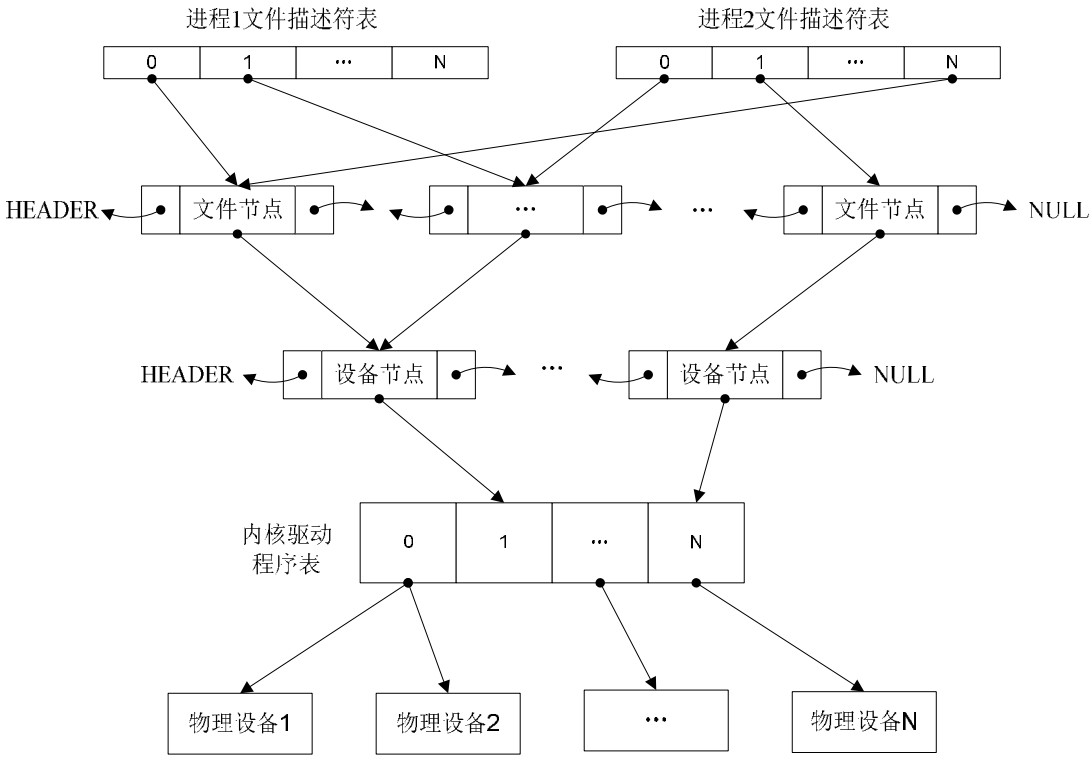
这里需要说明的是 SylixOS 支持 POSIX 规定的信号排队功能。

I/O 系统

SylixOS 兼容标准的 POSIX 输入输出系统，使用户非常容易上手。SylixOS 的 I/O 概念与 UNIX 兼容系统相同，认为一切都是文件。这些文件有不同的类型，它们包括：

1. 普通数据文件：最常用的文件类型，内部存放数据集合。
2. 目录文件：这个文件包含了其他文件的名字以及这些文件的指针。
3. 块设备文件：这种文件提供的 I/O 接口标准符合 SylixOS 对块设备的定义。
4. 字符设备文件：这是标准的不带缓冲的设备文件，系统中的设备不是块设备就是字符设备。
5. FIFO 文件：管道通信文件。它对应了一个命名管道。
6. 套接字（socket）文件：进程间或者主机间的网络通信。
7. 符号链接：该文件指向另一个文件。

SylixOS I/O 系统结构如下图所示。



不同的文件描述符可以对应同一个文件节点，当对应同一个文件节点的所有文件描述符被关闭则操作系统会释放对应的文件节点，同时调用相应的驱动程序。不同的文件节点可以指向同一个逻辑设备，例如一个 FAT 文件系统设备就可以被打开很多个文件节点。不同的逻辑设备也可以对应一个驱动程序，例如串口 0、串口 1 可以对应一组为其服务的驱动程序。当然哪一组驱动程序为哪一个硬件服务则由用户底层 BSP 代码决定。

文件描述符

SylixOS 的文件描述符与 POSIX 定义兼容，它是从 0 开始一直到一个最大值的整型数字，每一个打开的文件都有一个或者多个（dup）文件描述符与之对应。SylixOS 和绝大多数操作系统相同，打开文件时总是使用一个最小的未使用的文件描述符作为新分配的文件描述符。

根据习惯 0 号文件描述符代表标准输入，即 scanf 对应的输入文件，一般为一个终端，1 号文件为标准输出，即 printf 对应的输出文件，2 号文件为标准错误，即 strerror 对应的输出文件。这里需要说明的是 SylixOS 每个进程拥有自己的文件描述符表。各个进程间互不冲突，子进程继承父进程除 FD_CLOEXEC 外的所有文件描述符。一个进程内的所有线程共享进程文件描述符。而内核中也有一个全局的文件描述符表，这张表不包含 0，1，2 号标准文件，这三个文件描述符在内核中为重映射标志，即 SylixOS 允许内核中每个内核任务拥有自己的标准文件。

针对于文件操作，SylixOS 提供一组统一的 API 函数。这组 API 分为三类，他们是标准 I/O 操作、异步 I/O 操作和高级 I/O 操作。

标准 I/O

标准 I/O 又称作同步 I/O 操作，SylixOS 支持 POSIX 规定的绝大多数同步输入输出操作，

他们分别是：creat、open、close、unlink、read、write、readv、writev、pread、pwrite、pread64、pwrite64、ioctl、stat、stat64、lstat、lstat64、lseek、lseek64、fsstat、vfsstat、dup、dup2、ttyname、sync、fsync、fdatasync、fchmod、chmod 等等。这些 API 的行为与 POSIX 规定的一致，所以其他 POSIX 兼容操作系统的应用程序只需重新编译就可以在 SylixOS 系统上运行。

以上这些 API 都属于同步 I/O 范畴，即发起传输和对 I/O 的控制都是用户主动行为。设备必须在用户的干预下运转，目前绝大多数应用软件都使用这一类型的 I/O 操作。

异步 I/O

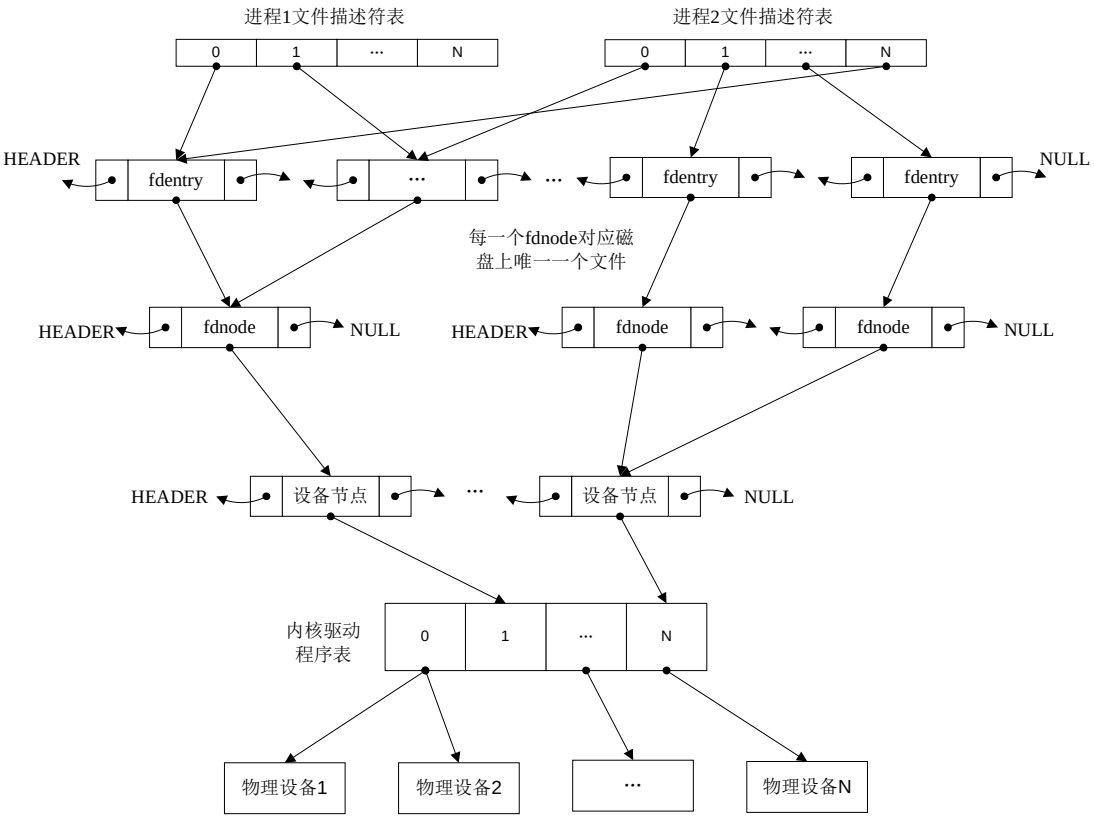
SylixOS 支持 POSIX1003.1b 实时扩展协议规定的标准异步 I/O 接口，即：aio_cancel、aio_error、aio_fsync、aio_read、aio_write、aio_return、aio_suspend、lio_listio。这组 API 用来操作异步 I/O。异步 I/O 是针对同步 I/O 提出的概念，它不需要线程等待 I/O 处理结果，而只需要请求进行传输，然后系统会自动完成 I/O 传输，结束或者出现错误时会产生相应的 I/O 信号，用户程序只需要设置好对应的信号陷入函数，即可处理一个异步 I/O 事件。

高级 I/O

POSIX 系统高级 I/O 是针对于标准 I/O 的不足进行设计的。这些功能包括：非阻塞 I/O、记录锁、I/O 多路复用、存储器 I/O 映射。

非阻塞 I/O 用于可能产生阻塞的 I/O 操作，例如：read、wirte 等等。用户可以通过相关 API 设置指定的文件描述符 O_NONBLOCK 标志，将对应的文件设置为非阻塞 I/O 模式，这时如果设备没有准备好，则系统调用会立即退出不会发生阻塞，同时 errno 将设置为 EWOULDBLOCK，表示对应的操作没有完成，需要稍后再重新执行。用户设置 O_NONBLOCK 标志的 API 包括 ioctl 的 FIONBIO 命令或者 fcntl 的 F_SETFL 选项。

记录锁的功能是：当一个进程正在读取或者修改文件的某个部分，他可以阻止其它进程修改同一文件的相同区域，它可以用来锁定文件的某个区域或者这个文件，SylixOS 支持文件记录锁的 API 是 POSIX 标准的 fcntl 和 BSD 系统的 flock 函数，SylixOS 同时兼容 BSD 系统在文件打开阶段自动加锁，只需要在 open 调用时加入 O_SHLOCK 与 O_EXLOCK 参数即可。SylixOS 中分为多种设备驱动模型但是只有 NEW1 型设备驱动支持记录锁功能，例如 FAT、NFS、YAFFS、ROMFS 等常用文件系统都是用这种驱动模型，它类似于 UNIX 系统的 vnode 类型，使用 NEW1 设备驱动的 I/O 系统结构如下图所示。



I/O 多路复用可以使线程阻塞在多个 I/O 请求上并且设置超时时间，当等待多个 I/O 请求时，如果有某一个文件描述符对应的事件允许时，阻塞的线程立即被激活，然后通过相关 API 判断有哪几个文件产生了相关的事件。它允许用户同时处理多个 I/O 的传输请求。I/O 多路复用的 API 包括：select 和 poll。SylixOS 同时支持这两个 API。

存储器 I/O 映射可以使对一个文件的操作变成对内存的操作，它使一个文件和内存中的一个缓冲区进行映射，对缓冲区内存的操作相当于对文件的操作。这个操作需要 SylixOS 的虚拟内存管理支持，操作系统将通过缺页中断系统为这个映射的文件服务。完成存储器 I/O 映射的 API 是 mmap。它不仅能够映射普通文件到内存，还可以映射设备文件（当然需要设备驱动程序支持），例如：我们在嵌入式系统中常用的显示设备是 framebuffer，这时可以将这个设备（通常文件名为/dev/fb0）通过 mmap 映射到虚拟空间，这时对映射空间的操作实际就是操作显存本身。

标准设备

上一节介绍的是 SylixOS 输入输出系统对用户提供的 API 基本内容，本节介绍 SylixOS 在 I/O 层之下提供的基本设备模型，这些设备包括普通设备类和总线型设备类：

普通设备类是 SylixOS 对一些标准设备行为的统一抽象，这些设备包括：

1. 终端
2. 虚拟终端
3. CAN 节点
4. 管道
5. 共享内存
6. DMA
7. 键盘鼠标

8. 实时时钟
9. 简单图形设备
10. 声卡
11. 块设备
12. 随机数发生器设备
13. ATA（SATA）设备
14. 空设备

……

用户驱动可以使用这些标准化的设备模型来编写，这样可以对上层提供统一的，标准的设备 API，方便应用程序移植。

同时 SylixOS 还提供了标准的总线型设备传输支持，他们包括：

1. SD 总线
2. SPI 总线
3. IIC 总线
4. USB 总线

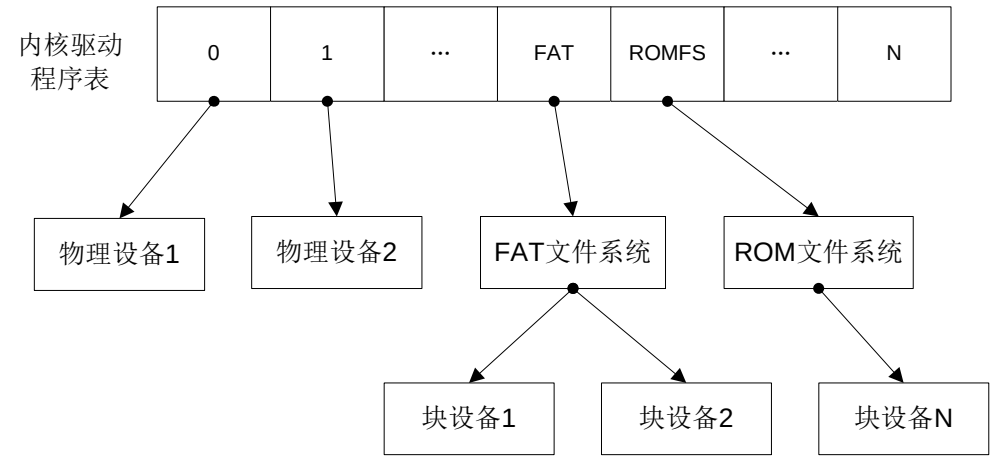
……

其中 PCI 和 USB 总线的支持作为 SylixOS 的内核模块在运行时动态装入。这些总线设备使用 SylixOS 统一的总线管理器确保总线上的设备传输互不受影响。

内建文件系统

SylixOS 内部提供了多种标准的文件系统，方便用户使用，这些文件系统是 SylixOS 编译内核时内建的，如果需要更多的文件系统如：NTFS 则需要通过内核模块运行时动态加入。

SylixOS 的文件系统实际上就是一组虚拟的设备驱动，它提供两组 API 接口，对上符合 I/O 系统虚拟文件系统（VFS）标准，对下要求设备符合块设备标准。文件系统在 I/O 系统中的结构如下图所示。



SylixOS 文件系统使用 I/O 系统提供的标准 VFS 进行挂载，然后通过标准 I/O 操作函数进行访问，换句话说，操作一个普通文件与操作一个设备文件没有什么不同。

SylixOS 目前内建的文件系统包括：

1. 根文件系统
2. PROC 文件系统
3. FAT 文件系统
4. YAFFS 文件系统

5. NFS 文件系统
6. ROMFS 文件系统

其中 ROMFS 文件系统是只读文件系统，系统的关键性文件都可以放在此文件系统中确保安全。如果通过 mount 挂载文件系统则 FAT、NFS，YAFFS 也都可以挂载成只读文件系统。

PROC 文件系统是保存操作系统信息和进程信息的文件系统，这个文件系统对应的文件实体都在操作系统内核中，是内核反馈出来的运行参数和信息，例如每一个进程的进程号都是一个对应的文件夹，里面存放着进程当前运行的信息，例如：进程对应的可执行文件，进程打开的文件描述符表，进程消耗的内存信息，进程内部的动态链接库信息等等。

SylixOS 内部所有的设备（包括文件系统）都必须挂载到根文件系统上，根文件系统的设备名称非常特殊，为：“/”。所有的设备或者文件绝对路径都是以根符号起始的，也就是说操作系统查询一个设备总是从根文件系统开始的。

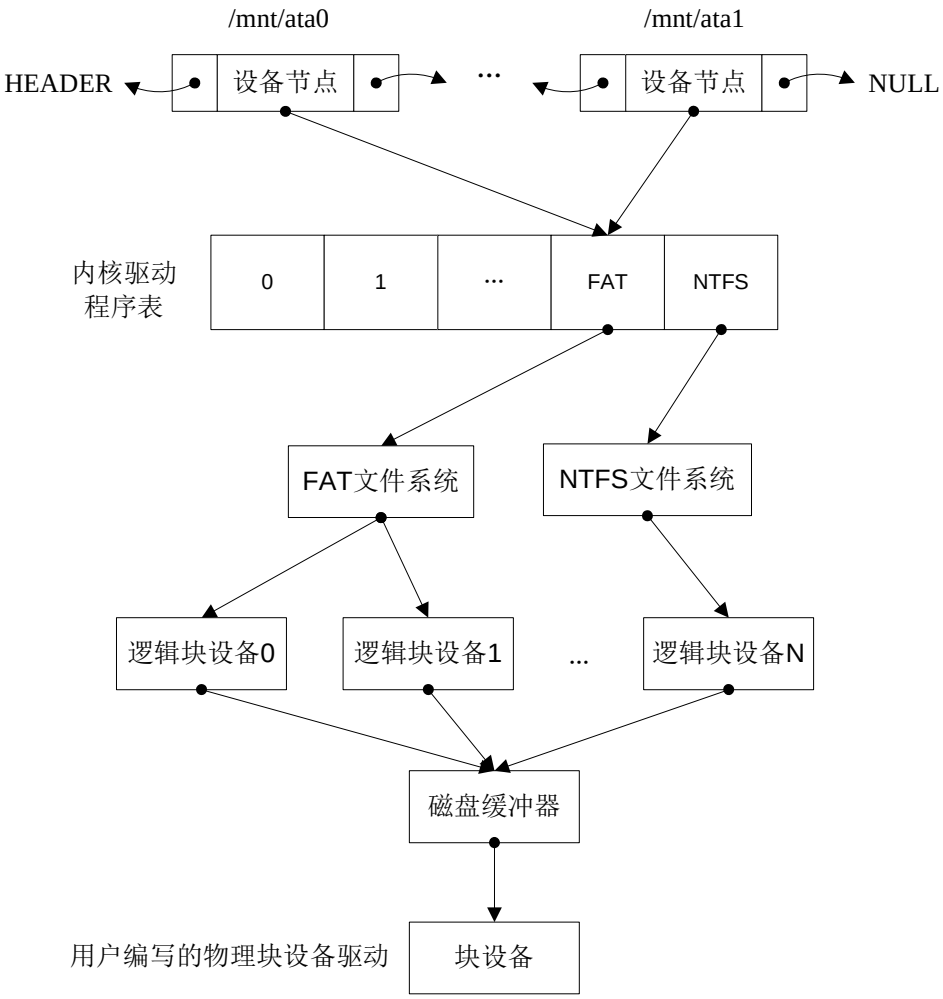
SylixOS 还提供了一些方便文件系统使用的组件，它们包括：磁盘分区检查工具，磁盘缓冲器，磁盘自动挂载工具等等。

其中磁盘分区检查工具可以自动的检查一个磁盘的分区情况，并且生成对应分区的逻辑设备，每个逻辑设备都可以进行文件系统挂载。

磁盘缓冲器是一个特殊的块设备，他可以介于文件系统和磁盘之间，由于磁盘是低速设备，磁盘的读写速度远远低于 CPU，所以为了解决这种速度不匹配，SylixOS 提供了对应块设备的缓冲器。它可以极大的减少磁盘 I/O 的访问率，同时提高系统性能。当然引入磁盘缓冲器的原理同 CPU 的 CACHE 一样，所以它会带来存储器与磁盘中的数据短时间内不一致的现象，这个问题可以通过 sync、fsync 或者 fdatasync 函数调用来同步数据。其中 sync 将阻塞当前线程，然后将系统中需要从缓存中回写的数据全部写入到设备中再返回。fsync 表示将指定文件的缓存数据同步到物理磁盘中，fdatasync 表示将指定文件的数据部分同步写入物理磁盘。

磁盘自动挂载工具是将很多磁盘工具封装在一起的一个工具集。设备可以通过热插拔事件将物理磁盘块设备交给磁盘自动挂载工具，这个工具首先会为这个磁盘开辟磁盘缓冲，然后会自动进行磁盘分区检查，最后生成对应每个分区的虚拟块设备，最后这个工具会识别每一个分区的文件系统类型，并装载与之对应的文件系统，这样从用户角度来说，就可以在操作系统目录中看到对应挂载的文件系统目录了。

带有磁盘缓冲器和分区处理工具的 SylixOS 块设备结构如下图所示。



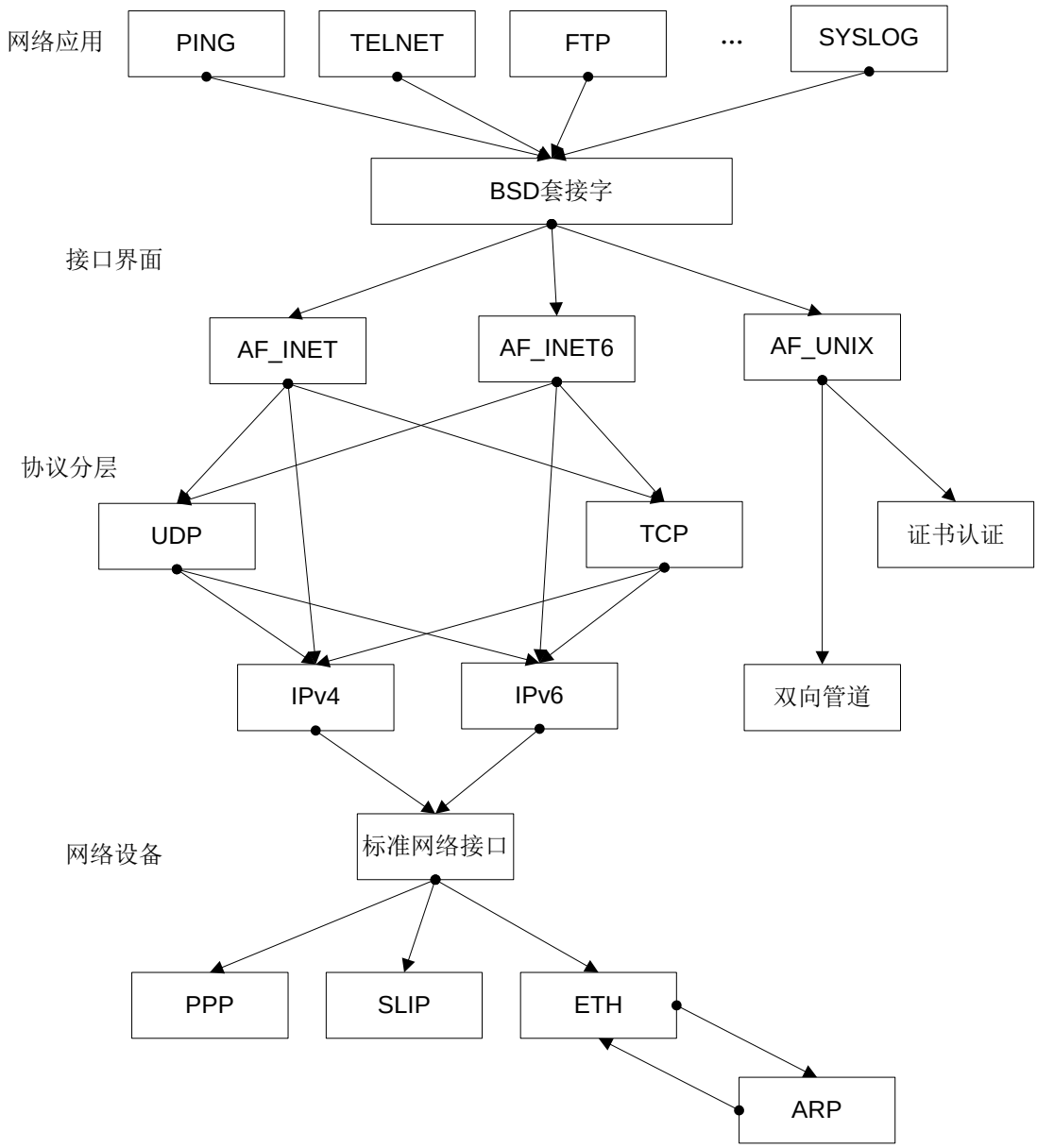
网络通信

SylixOS 支持 BSD 标准的 socket 通信接口，这些接口包括：socketpair、socket、accept、bind、shutdown、connect、getsockname、getpeername、setsockopt、getsockopt、listen、recv、recvfrom、recvmsg、send、sendto、sendmsg 等等，同时由于 socket 也是 I/O 系统的一部分，所以它支持 read、write、ioctl、select、fcntl 等等标准 I/O 的操作 API。

SylixOS 的 socket 支持四类域它们分别是：

1. AF_INET　　IPv4 协议
2. AF_INET6　　IPv6 协议
3. AF_UNIX　　UNIX 域协议（socketpair 仅支持 AF_UNIX 域）
4. AF_PACKET　链路层数据包收发接口

SylixOS 系统网络系统构架如下图所示。



AF_INET 和 AF_INET6 可以用作互联网通信，例如浏览网页，发送电子邮件，传输文件，网络可视电话等等。

在 AF_INET 和 AF_INET6 下支持的 socket 类型有：SOCK_STREAM、SOCK_DGRAM 和 SOCK_RAW，他们分别称作：流式套接字，用户数据报套接字和原始套接字。其中流式套接字使用 TCP 协议进行通信，它提供面向连接的，可靠的全双工管道通信。用户数据报套接字使用 UDP 协议进行通信，它提供非连接的，不可靠的通信。原始套接字可以让用户自行操作一些更加底层的通信，例如 ICMP 数据包等。

AF_UNIX 可以用作本机进程间双向管道通信，例如在 POSIX 系统中大多数支持多进程的图形系统，进程间都是使用 AF_UNIX 进行双向管道通信。最为著名的是 X windows(X11 或 X)系统，这个系统使用非常广泛。在它之上运行着很多大型的图形界面，例如：GTK+。同样在嵌入式系统里面 Qt 的 QWS 也是借助 AF_UNIX 实现多进程间通信的。

进程模型

SylixOS 在版本 1.0.0 时引入了进程概念，成为全世界少数几个支持进程的实时操作系统之一。在以往的嵌入式系统中大多数系统不存在 MMU，CPU 性能也很低，所以根本没有引入进程的必要性，只需要一个多线程操作系统就可以工作的非常好，可是时过境迁，如今的嵌入式处理器性能普遍较高，而且都含有完整的 MMU，这使得嵌入式系统在能接受的复杂度下支持多进程成为可能。

由于应用场合与 PC 大为不同，所以面向工业领域的嵌入式系统大多数为实时操作系统，所以为了实现实时性特点，SylixOS 引入的进程类似于 VxWorks6.0 之后的 RTP 进程与 uClinux 的进程，也就是说 SylixOS 内所有进程共享一个地址空间，只是在动态链接的时候进行代码和数据的重定向，那么进程调度时就不需要切换庞大的页表，这样进程的切换时间就符合实时系统对时间确定性的要求。SylixOS 进程实际上是一个资源的容器，这个容器中包含代码段，数据段，文件描述符，内核对象，posix 对象，C 库和众多支持的动态链接库，这些库和应用程序共享一个进程容器，进程容器间彼此隔离，那么在使用起来和 linux 上的标准进程也就没有任何区别了。

进程标准

SylixOS 支持 POSIX 除 fork 以外的所有进程 API，因为没有 fork，所以 SylixOS 创建进程可以使用 ISO C 规定的 spawn 系列函数（头文件为 process.h）。同时 SylixOS 支持 POSIX1003.1b 规定的实时进程标准，这些函数在<spawn.h>中声明。这些函数可以满足用户对进程的绝大多数需求。

进程资源回收

SylixOS 进程分为三种类型，它们分别是：正常进程、孤儿进程、僵尸进程。正常进程是指存在父亲进程且本进程与父进程都在运行的进程。孤儿进程是指父进程先于当前子进程结束，当前子进程还在运行的进程。僵尸进程是指子进程已经结束，正在等待父进程回收自己的资源，可是父进程退出时没有回收子进程资源。

对于**正常进程**，在结束时需要父进程调用 wait 或者 waitpid 来回收自己的资源，那么资源的回收将在父进程中进行。

对于**孤儿进程**，SylixOS 内核线程 t_reclaim 将接管这个进程，该进程结束时，t_reclaim 将回收这个进程的所有资源，t_reclaim 内核线程类似于 Linux 系统的 init 进程。

对于**僵尸进程**，由于父进程没有调用 wait 或者 waitpid 回收子进程就直接退出，这时操作系统 t_reclaim 内核线程将会自动接管该进程的所有子进程，同时回收该进程所有已经退出但没有回收资源的子进程的资源。这种算法可以使 SylixOS 尽可能的减少僵尸进程对系统资源的浪费。

进程间通信

SylixOS 提供了较为丰富的进程间通信方法，分为异步通信和同步通信。其中同步通信都是采用文件方式，这些文件都存在于内核内存中。

进程间异步通信可以直接使用操作系统提供的信号接口，通过 kill 或者 sigqueue 系统调用发送指定的信号给接收进程。

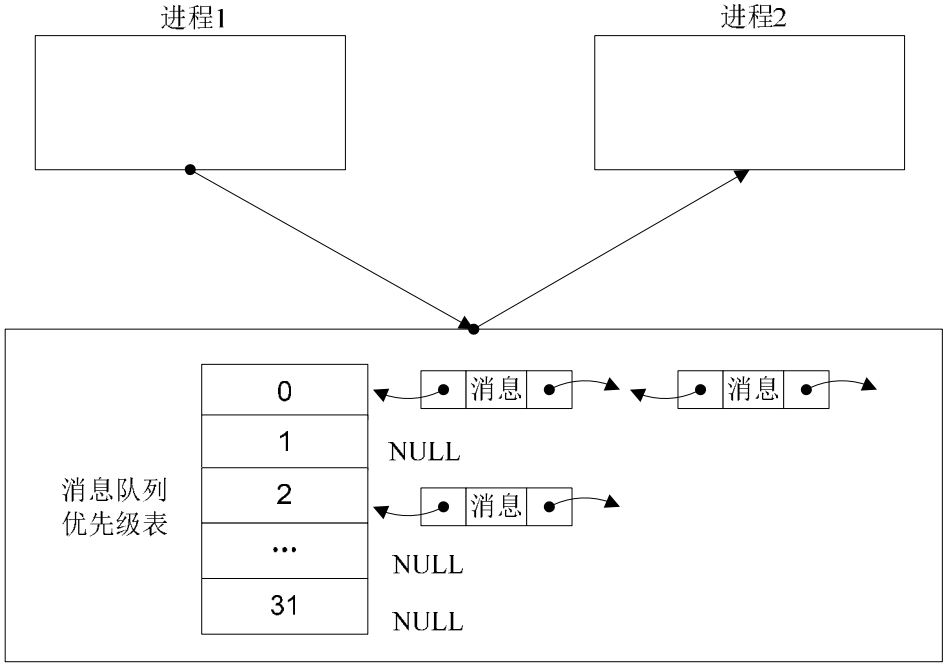
进程间同步通信方式比较多，他们包括：

1. POSIX 命名信号量
2. POSIX 命名消息队列
3. 命名或匿名管道
4. socket 套接字管道
5. 共享内存

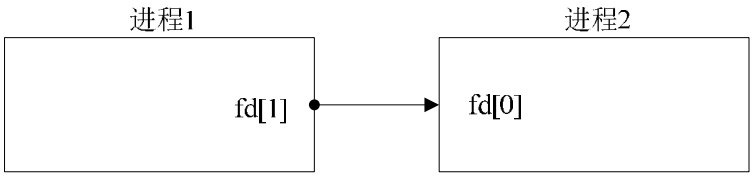
注意：POSIX 命名对象是虚拟存在的，它并不对应操作系统 I/O 空间内的实体文件，所以，在操作系统 I/O 空间内并不显示，用户可以通过查询/proc/posix/pnamed 文件来获取操作系统中所有 POSIX 命名对象的信息。

POSIX 命名信号量：多个进程可以通过 sem_open 创建或者打开拥有同样名称的命名信号量，用此信号量进行同步通信。

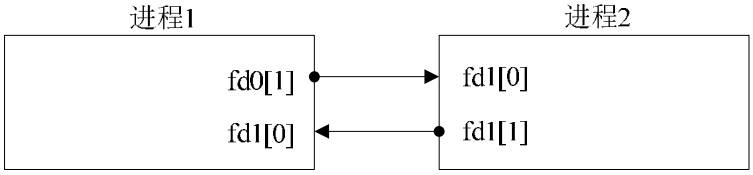
POSIX 命名消息队列：多个进程可以通过 mq_open 创建或者打开拥有同样名称的命名消息队列，用此消息队列进行同步通信。需要说明的是 SylixOS 支持的 POSIX 消息队列拥有 32 个消息优先级，消息根据优先级在消息队列内排序，优先级越高的消息最先被接收。下图为进程间使用消息队列通信示意图。



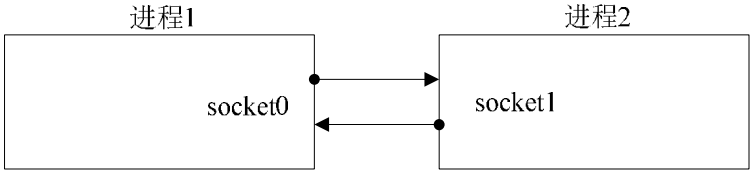
命名或匿名管道：命名管道通过 mkfifo 来创建，其它进程可以打开这个管道进行进程间通信。匿名管道通过 pipe 创建，pipe 将返回两个文件描述符，一个是读端，另一个是写端，子进程可以通过继承父进程的描述符，也可通过 POSIX 实时进程提供的 file_action 参数传递文件描述符进行进程间通信。下图为进程间使用管道通信示意图。



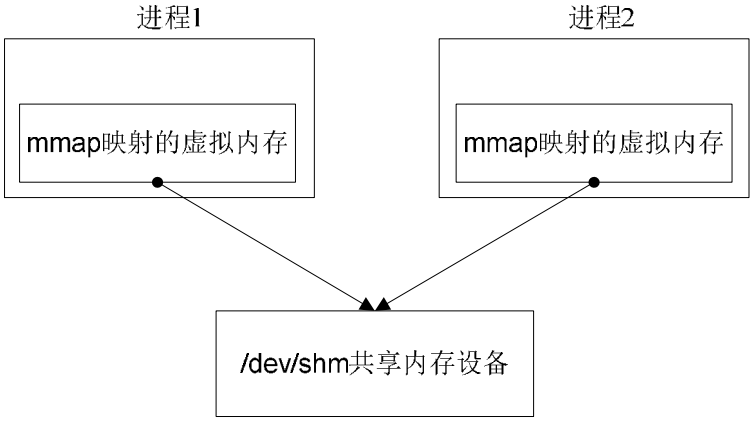
如果进程间需要双向通信，则可以建立两条管道，如下图所示。



socket 套接字管道：多个进程间可以使用 AF_UNIX 协议域进行通信，需要说明的是 socket 套接字管道不同于标准命名或匿名的单向数据流管道，socket 套接字管道不同于普通的单向数据流管道，它是双向全双工管道。当然用户也可以通过 TCP/IP 的 127.0.0.1 回环地址进行通信。下图为进程间使用 socket 管道通信示意图。



共享内存：SylixOS 提供专用的共享内存设备/dev/shm，此设备为虚拟设备，它不同于 Linux 上的 tmpfs 文件系统，SylixOS 的共享内存设备只为了共享内存服务。换句话说所有对该设备的内存映射操作，虽然返回的虚拟地址不同，但是物理地址是相同的。应用程序可在此设备上创建虚拟文件并设置虚拟文件的大小，不同的进程可使用 mmap 函数映射相关的文件内存实现进程间内存的共享。下图为进程间使用共享内存通信示意图。



内核模块与动态链接库

SylixOS 支持内核模块与动态链接库概念，SylixOS 的内核模块与 Linux 内核模块概念和功能基本相同，加入内核的内核模块，实际上就是内核的一部分，一般利用内核模块实现一些基本的内核服务，例如 USB 和 PCI 的总线抽象层；一些外设的驱动程序；额外要支持的文件系统等等。

SylixOS 内核可以编译成只带有少数驱动的镜像，然后根据运行硬件的不同，加入不同的内核模块（包含对应的硬件驱动程序）。SylixOS 内核模块与 Linux 格式相同，内核模块文件名一般为*.ko。内核模块一般放置于/lib/modules 目录下，设备驱动则一般放在/lib/modules/drivers 目录下。

SylixOS 支持应用程序动态链接库功能，一个进程内可以加入无限制数量的动态链接库，他们可以由操作系统自动加入也可以由用户代码手动加入。

自动加入动态库：开发应用程序时，在链接阶段通过链接命令指定进程运行时依赖的动

态链接库，并把相应的动态库放在环境变量 LD_LIBRARY_PATH 指定的目录中，这样进程启动时 SylixOS 会自动装载依赖的动态链接库到指定的进程空间中。

手动加入动态链接库：进程在运行时可以使用 POSIX 提供的动态链接库功能手动的装载动态链接库。这些 API 在头文件 dlfcn.h 中声明，它们分别是：dlclose、dlerror、dlopen、dlsym、dladdr。其中 dladdr 不是标准的 POSIX 函数，dladdr 提供从地址获取近似符号的功能。

提示：进程崩溃时（收到 SIGSEGV 等崩溃信号）相关信号句柄可以通过 execinfo 函数获取到当前的调用栈信息。然后通过 dladdr 来分析调用栈中的函数名称，这样有利于分析应用程序的 BUG。

SylixOS 动态链接库的格式也与 Linux 相同，动态链接库的文件名一般为*.so。如果存在一个动态库多个版本的情况，可以采取 Linux 对动态库版本的命名方法进行区分，同时建立一个符号链接文件指明当前默认使用的动态链接库。

C++支持

SylixOS 内核提供了一个小型的 C++运行时环境，内核支持运行基本的 C++程序，这些程序不能带有异常处理（exception）和运行时类型识别（rtti）。因为相关的操作需要完整的 C++运行时支持。

内核中的 C++程序分为两种：

1. 随 SylixOS 内核一起编译的 C++程序
2. SylixOS 装载含有 C++程序的内核模块

对于随 SylixOS 内核一起编译的 C++程序，系统将会引出四个符号：

__LW_CTOR_LIST__

__LW_CTOR_END__

__LW_DTOR_LIST__

__LW_DTOR_END__

这四个符号需要在 SylixOS 链接脚本中放在指定的位置，用他们来指明内核 C++程序的全局构造函数表和全局析构函数表。SylixOS 在启动时会运行相关的 C++全局构造函数表，SylixOS 在结束时也会运行相应的全局析构函数表。全局构造与析构函数表如下图所示。

操作系统内核全局构造函数表

__LW_CTOR_LIST__
Constructor[0]
Constructor[1]
Constructor[2]
…
Constructor[N]
__LW_CTOR_END__

操作系统内核全局析构函数表

__LW_DTOR_LIST__
Destructor[0]
Destructor[1]
Destructor[2]
…
Destructor[N]
__LW_DTOR_END__

对应的连接脚本范例如下所示。

```
.ctors :
{
```

```
        KEEP (*cppRtBegin*.o(.ctors))
        KEEP (*(.preinit_array))
        KEEP (*(.init_array))
        KEEP (*(SORT(.ctors.*)))
        KEEP (*(.ctors))
        KEEP (*cppRtEnd*.o(.ctors))
    }
    .dtors :
    {
        KEEP (*cppRtBegin*.o(.dtors))
        KEEP (*(.fini_array))
        KEEP (*(SORT(.dtors.*)))
        KEEP (*(.dtors))
        KEEP (*cppRtEnd*.o(.dtors))
    }
```

相关的文档见 SylixOS doc 目录的 CPLUSPLUS 文件。

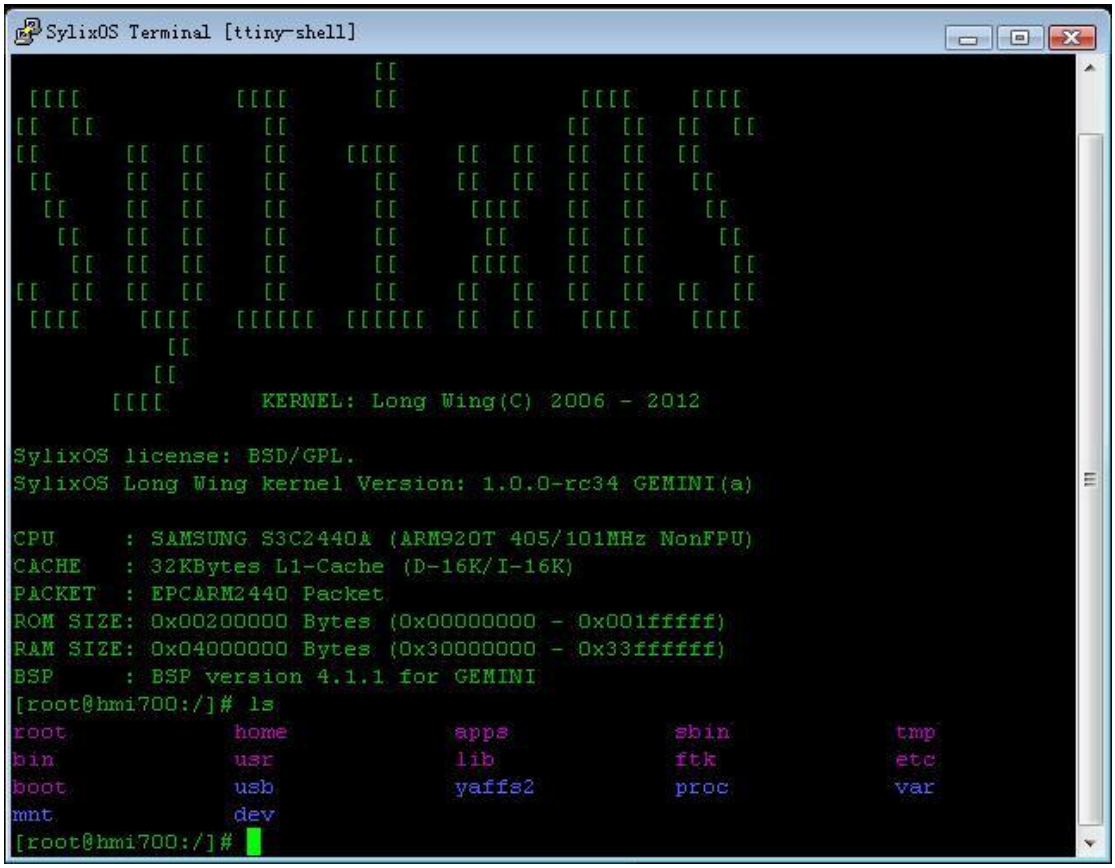
SylixOS 装载含有 C++程序的内核模块：SylixOS 装载器会自动的识别全局构造函数和析构函数表，然后在适当的时机运行。

应用程序中如果使用 C++程序则完全没有内核中使用的诸多限制，因为 C++程序可以链接编译器提供的标准 stdc++库，这个库和 SylixOS 内部的 C++基础库同时构成完整的 C++运行时库，能够支持异常与运行时类型识别等 C++高级特性。

内核 Shell

为了方便调试，SylixOS 提供一个类似 Linux Shell 终端的超小型终端（ttiny-shell，在 tiny 前再加入一个 t 表明比 tiny 还小），它运行在内核中，除了完成基本的 shell 功能，还能检测并设置内核的诸多参数。需要说明的是 shell 为内核线程，并非 Linux 下的 sh 进程，所以由 shell 启动的进程没有父亲，从进程开始运行时他就是孤儿进程，操作系统通过 t_reclaim 内核线程回收其资源。

Shell 的基本运行界面如下图所示：


SylixOS Terminal [ttiny-shell]
KERNEL: Long Wing(C) 2006 - 2012
SylixOS license: BSD/GPL.
SylixOS Long Wing kernel Version: 1.0.0-rc34 GEMINI(a)
CPU : SAMSUNG S3C2440A (ARM920T 405/101MHz NonFPU)
CACHE : 32KBytes L1-Cache (D-16K/I-16K)
PACKET : EPCARM2440 Packet
ROM SIZE: 0x00200000 Bytes (0x00000000 - 0x001fffff)
RAM SIZE: 0x04000000 Bytes (0x30000000 - 0x33ffffff)
BSP : BSP version 4.1.1 for GEMINI
[root@hmi700:/]# ls
root home apps sbin tmp
bin usr lib ftk etc
boot usb yaffs2 proc var
mnt dev
[root@hmi700:/]#