参考资料：

https://www.cnblogs.com/jiduoduo/p/14401558.html

https://www.cnblogs.com/MrHCD/p/14651332.html

https://www.cnblogs.com/dahuige/p/15175137.html

POSIX表示可移植操作系统接口（Portable Operating System Interface of UNIX，缩写为 POSIX ），POSIX标准定义了操作系统应该为应用程序提供的接口标准。

POSIX标准意在期望获得源代码级别的软件可移植性。换句话说，为一个POSIX兼容的操作系统编写的程序，应该可以在任何其它的POSIX操作系统（即使是来自另一个厂商）上编译执行。简单总结：

完成同一功能，不同内核提供的系统调用（也就是一个函数）是不同的，例如创建进程，linux下是fork函数，windows下是creatprocess函数。好，我现在在linux下写一个程序，用到fork函数，那么这个程序该怎么往windows上移植？我需要把源代码里的fork通通改成creatprocess，然后重新编译...

posix标准的出现就是为了解决这个问题。linux和windows都要实现基本的posix标准，linux把fork函数封装成posix\_fork（随便说的），windows把creatprocess函数也封装成posix\_fork，都声明在unistd.h里。这样，程序员编写普通应用时候，只用包含unistd.h，调用posix\_fork函数，程序就在源代码级别可移植了。

1、简介

POSIX（The Portable Operating System Interface）是一个IEEE标准，有助于操作系统之间的兼容性和可移植性。从理论上说，符合POSIX的源代码应该可以无缝移植。在现实世界中，应用程序转换通常会遇到系统特定的问题。但是POSIX规范性使得移植应用程序更加简单，这可以节省时间。因此，开发人员应该熟悉这个广泛使用的标准的基础内容。

2、POSIX的历史

早期的程序员必须为每一种新的计算机模型从头开始重写其应用程序。但是IBM System/360改变了这一点。1964年，它推出了操作系统OS/360。IBM开始应用相同的硬件体系结构，以使得新模型能够重复使用相同的操作系统。多计算机平台的通用操作系统的存在为应用程序的可移植性设置了第一阶段。

在60年代后期，UNIX的出现开辟了新的可能性。AT&T的贝尔实验室负责此新操作系统的初始开发。它可以跨多个供应商的机器运行。同时UNIX开始涉足各种各样的领域。除了AT&T的System V外，还有Berkeley Software Distribution（BSD），XeniX等。移植这些种类并不容易。应用程序可移植性的承诺遇到了障碍。在未来几十年中，新操作系统的引入只会使跨硬件，操作系统和供应商的应用程序移植变得更加复杂。

POSIX标准是在1980年代开发的，旨在解决可移植性问题。该标准是基于System V和BSD Unix定义的。POSIX不定义操作系统，它仅定义应用程序和操作系统之间的接口。只要遵循两者之间的接口，程序员就可以随意编写自己的OS和应用程序。由于POSIX独立于硬件，操作系统或供应商，因此更容易实现应用程序的可移植性。

最初，POSIX被划分多个标准：

POSIX.1：Core Services

POSIX.1b：Real-Time extensions

POSIX.1c：Threads extensions

POSIX.2：Shell and Utilities

1997年之后，奥斯汀集团将所有标准归为一类。从那时起，版本POSIX.1-2001（也称为IEEE标准1003.1-2001），POSIX.1-2004（也称为IEEE标准1003.1-2004）和POSIX.1-2008（也称为IEEE标准1003.1-2008）.

某些POSIX兼容系统的例子有：AIX，HP-UX，Solaris和Mac OS。除此之外，Android，FreeBSD，Linux发行版，OpenBSD，VMWare等遵循大多数POSIX标准，但未经过认证。

3、POSIX基础

POSIX.1-2008标准涉及四个主要领域：

1）Base Definition Volume：通用术语、概念和接口。

2）System Interfaces Volume：系统服务功能和子例程的定义。此外，还包括可移植性，错误处理和错误恢复。

3）Shell and Utilities Volume：定义任何应用程序到Command shell和通用Utilities的接口。

4）Rationale Volume：包含有关添加或放弃的功能以及决策依据的信息和历史记录。

该标准不包括图形界面，数据库界面，object/二进制代码的可移植性，系统配置，I/O注意事项或资源可用性。

4、POSIX涉及背后的一些指导原则是：

1）创建POSIX的目的是简化应用程序的可移植性。因此，它不仅适用于UNIX系统。非UNIX系统也可以兼容POSIX。

2）该标准并不决定应用程序或操作系统的开发。它仅定义了它们之间的联系。

3）符合POSIX的应用程序源代码应该能够在许多系统上运行，因为该标准是在源代码级别定义的。但是，该标准不保证任何Object/二进制代码的可移植性。因此，即使在具有相同硬件和操作系统的类似机器上，二进制执行文件也可能无法运行。该标准仅解决了源代码的可移植性。

4）POSIX是用标准C编写的。但是开发人员可以用他们喜欢的任何预言来实现它。

5）该标准仅处理与应用程序交互的操作系统的各个方面。

6）该标准在长度与范围上保持简洁，已覆盖大量系统。

7）POSIX旨在简化可移植性。因此从长远来看，它将节省时间和金钱。但是，如果您的应用程序不兼容POSIX，则一开始可能需要大量的时间和资源投资。

5、POSIX应用程序开发

POSIX的目的是提高可移植性。当你的源代码符合标准时，你可以轻松地在另一台计算机上编译和运行代码。但是，如果将POSIX定义为应用程序的一般需求，则可能引起混乱。完整的POSIX标准是4000多个页面，具有1350多个接口。全部实现是没有任何意义的。因此，每个项目都应定义满足特定需求的POSIX。

在开发者社区中有一个误区，认为POSIX标准过时且不可靠。这并不正确,POSIX是一份实时文档Austin Group会定期对其进行更新。任何人都可以加入该小组并参与提高标准。该标准在当今服务器，工作站，路由器，移动设备，嵌入式系统等使用很活跃。它用于UNIX和Linux计算机。

6、结论

POSIX乍一看似乎并不平易近人。尽管如此，应用程序开发人员应该熟悉基本知识，因为它会不时出现在需求中。由于标准范围广，不可能熟悉整个文档，按需学习就好。