Thema

Portable Operating System Interface (POSIX)

Universität Hamburg

Betreuer: Eugen Betke

Hausarbeit von: Singh Khattar, Johnvir

Abgabe am: 15.01.2019

Adr.: Hinter der Dorfkirche 57

Mail: johnvirsingh@hotmail.de

Studium – Fachrichtung: B.Sc.Wirtschaftsinformatik

Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Inhaltsverzeichnis

Abstract

POSIX ist eine Gruppe von Standards, die die Portabilität von Anwendungssoftware auf verschiedenen Betriebssystemen gewährleisten soll. Diese Arbeit behandelt das Thema POSIX und beschreibt die Inhalte dieser Spezifikation. Außerdem befasst sich diese Ausarbeitung mit einer kritischen Perspektive auf dieses Thema.

1 Einleitung

POSIX ist ein Akronym und steht für das Wort „Portable Operating System Interface“. Es beschreibt eine Gruppe von Standards, die Ende der 80er Jahre von der IEEE Organisation erstmals als einheitlicher Standard für Betriebssysteme in Form einer Programmierschnittstelle veröffentlicht wurde.[1]

In dieser Arbeit soll der POSIX-Standard (nachfolgend: POSIX) erarbeitet werden, indem zunächst die Spezifikation von POSIX erläutert wird. Dann soll sich mit der Fragestellung befasst werden, warum POSIX entwickelt wurde und welche Versionen es gab und welche Features diese Versionen jeweils beinhalten. Anschließend soll aus den generellen Informationen zur Spezifikation die Verwendung und die Relevanz von POSIX abgeleitet werden. Abschießend folgt eine kritische Betrachtung auf den POSIX Standard und ein Fazit, um die Inhalte dieser Arbeit nochmal zusammenzufassen.

Diese Arbeit soll primär einen Überblick über das Thema POSIX verschaffen und vereinzelt Einblicke in die Details dieser Spezifikation aufzeigen. Außerdem soll dem Leser das Thema POSIX aus verschiedenen Perspektiven aufgezeigt werden, indem eine kritische Betrachtung der Spezifikation vorgenommen wird.

2 Hintergrund der Arbeit

Zum Thema POSIX gibt es Ausarbeitungen und Veröffentlichungen beispielsweise in Fachzeitschriften, oder anderen Publikationen. Diese Arbeit beruht jedoch auf den offiziellen Veröffentlichungen der IEEE und The Open Group. Diese beiden Organisationen haben diesen Standard entwickelt und haben daher Publikationen erstellt, auf denen diverse Informationen über diesen Standard zu finden sind.

[1] Vgl. IEEE & The Open Group: A Backgrounder on IEEE Std 1003.1, Homepage: http://www.opengroup.org/austin/papers/backgrounder.html, Letzter Aufruf: 14.01.19

Diese Arbeit soll jedoch nicht nur die Eigenschaften von POSIX darstellen, sondern diese auch aus einer kritischen Perspektive darstellen und historisch und technisch einordnen. Dies soll auf Basis Fachdiskussionen und Veröffentlichungen anderer Organisationen und Experten erfolgen.

3 Hauptteil

3.1 POSIX – Entwicklung: Organisationen & Argumente

POSIX beschreibt eine Familie von Standards, die erstmalig von der IEEE veröffentlicht wurde. IEEE steht für Institute of Electrical and Electronics Engineers und ist ein Berufsverband von Ingenieuren. Dieses Institut hat erstmalig den IEEE Std 1003.1 entwickelt und herausgegeben. Dieser Standard stellt die erste POSIX-Spezifikation dar. [2]

Die POSIX-Spezifikation wurde wie eingangs erwähnt Ende der 80 Jahre entwickelt, da es zu dieser Zeit diverse Betriebssystemhersteller gab, die Ihre eigenen Entwicklungen autonom voneinander vorantrieben. Unter den Betriebssystemen zu dieser Zeit gab es vermehrt Linux-, oder Unix-basierte Systeme. Aufgrund der autonomen Weiterentwicklung der Betriebssysteme kam es zu dem Szenario, dass Softwarehersteller spezielle Implementationen in ihrem Source Code vornehmen mussten, damit die Kompatibilität der Anwendung gewährleistet ist. Im Fall, dass der Anwendungssoftwarehersteller eine neue Zielgruppe mit seinem Programm erreichen wollte, die ein anderes Betriebssystem verwendete, mussten in solch einen Fall Implementationen vollzogen werden, damit die neue Zielgruppe erreicht werden kann.
Es wurde sich daher um ein Standard bemüht, damit dieser Fall in dieser Form nicht wieder auftritt. So hat sich das IEEE damit befasst eine API (Abk. für Application Programming Interface) zu definieren. Eine API beschreibt eine Programmierschnittstelle, die verwendet

[2] Vgl. IEEE: Homepage der IEEE, https://www.ieee.org, Letzter Aufruf: 14.01.19

wird um gewisse Implementation bei anderen Programmen vorauszusetzen, damit diese sich an das Softwaresystem anbinden können.[3] Im Fall von POSIX hat das IEEE eine API definiert, welche die Betriebssystemhersteller implementieren können, so dass die Kompatibilität von Anwendungssoftware auf dem jeweiligen Betriebssystem gewährleistet ist. Implementiert solch ein Betriebssystem diese API, so bezeichnet man dieses als POSIX konformes Betriebssystem. Ist ein Betriebssystem POSIX konform, so bietet es den Softwareherstellern ein Standardverhalten an, so dass die Softwarehersteller sich nicht um die Kompatibilität der Anwendungssoftware bemühen müssen.

Bei der weiteren Entwicklung vom POSIX-Standard waren und sind noch weitere Institutionen beteiligt. The Open Group hat die Entwicklungen ab 1997 mit vorangetrieben, als die Austin Group sich geformt hat. Die Austin Group besteht aus dem IEEE, der The Open Group und aus der ISO/IEC JTC. So wurde der Standard auch unter einer ISO-Norm „ISO/IEC/IEEE 9945“ definiert.

3.1.1 POSIX – Spezifikation & Beispielauszüge

Damit die Inhalte von POSIX dargestellt werden, sollen im nächsten Abschnitt die Spezifikation beschrieben und Beispielauszüge und Beispielinhalte aufgezeigt werden.

3.1.1.1 Spezifikation

Im Punkt 3.1 wurde bereits dargestellt, warum der POSIX-Standard definiert wurde. Im Folgenden sollen der Aufbau der Spezifikation aufgezeigt und die Inhaltsstruktur dargestellt werden.

Die POSIX Spezifikation ist gegliedert in vier grobe Teile. Der erste Teil beschreibt Basis-Definitionen. In diesem Teil werden Konventionen, Konzepte und allgemeine Ausdrücke beschrieben und definiert. Der zweite Teil beschreibt die Systemschnittstelle: In diesem Teil

[3] Vgl. Gründerszene.de: Application-Programming-Interface, https://www.gruenderszene.de/lexikon/begriffe/application-programming-interface-api?interstitial, Letzter Aufruf: 14.01.19

werden Header von Funktionen definiert, wodurch ein Standardverhalten des implementierenden Systems vorgegeben wird. Es sind die C-Systemaufrufe aufgezeigt und es wird erklärt, wie beispielsweise auf Errors reagiert wird.
Im dritten Teil werden die „Shell & Utilities präsentiert“. Hierbei wird auf Quellcodeebene eine Standardschnittstelle definiert und es werden Hilfsprogramme für die Anwendungsprogramme aufgezeigt.
Im vierten und letzten Teil wurden Inhalte aufgegriffen, die nicht in den ersten drei Teilen verarbeitet werden konnten, wie historische Informationen, oder Erklärungen zu den definierten Standards in den ersten drei Abschnitten.[4]

Bei der Erstellung dieser Standards hat sich die IEEE an Prinzipien gehalten, die das Verständnis von der POSIX-Spezifikation und den darin enthalten Entscheidungen bezüglich Konventionen, Definitionen, etc. erleichtern sollen. Nachfolgend soll eine Auswahl dieser Prinzipen dargestellt werden.

Bei Erstellung des POSIX Standards wurde Anwendungsorientiert gearbeitet, mit dem Ziel, dass die Anwendungssoftware kompatibel ist und die primäre Zielsetzung von POSIX eingehalten wird. Mit diesem Prinzip folgte ein weiteres, welches auf dieses aufbaut, nämlich, dass minimale Änderungen an bestehender Anwendungssoftware notwendig sein sollen, sobald der Standard definiert ist. Weiterhin wurde angestrebt ein minimales Interface in diesem Standard zu definieren, ohne Implementation, um bei den Betriebssystemherstellern Flexibilität im vorgesehenen Rahmen zu schaffen. Es wurde außerdem die Programmiersprache Standard C zu Grunde gelegt, spezifiziert im ISO C Standard. Ein weiteres Prinzip war, dass der Standard über viele Betriebssysteme und Systeme hinweg anwendbar sein sollte. Hierbei wurde darauf geachtet, dass Systeme wie distributed systems, oder auch wie networked systems gleichermaßen POSIX

[4] Vgl. IEEE & The Open Group: The Open Group Base Specifications Issue 7, Publikation: http://pubs.opengroup.org/onlinepubs/9699919799.2016edition/, Letzter Aufruf: 14.01.19

konform sein können und der Standard viele dieser Systeme abdeckt. [5]

3.1.1.2 Beispielauszüge & -inhalte

In diesem Abschnitt sollen einige Beispielinhalte des POSIX Standards aufgezeigt und erläutert werden, wobei der Umfang dieser Ausarbeitung es nicht zulässt, dass detailliert auf die jeweiligen Themen eingegangen werden kann.

In der POSIX Spezifikation werden POSIX Threads (Nachfolgend: Pthreads) definiert. Diese werden von den meisten modernen Betriebssystemen unterstützt. Diese Reichweite von Pthreads begründet sich in der Tatsache, dass diese Prozeduren weniger Betriebssystemressourcen verbrauchen. Dies sorgt dafür, dass mehrere Threads gleichzeitig möglich sind und es steigert die Programmperformance, was das primäre Ziel beim Einsatz der Pthreads ist.[6] Grundlegende Funktionen von Pthreads sind pthread_create(), womit sich POSIX Threads erzeugen lassen und pthread_join(), wodurch diese sich zusammenführen lassen. [7]

Mit dem Standard lassen sich POSIX Prozesse definieren. Diese stellen mit dem Befehl fork() eine Alternative zu den Pthreads dar. Anhand der Abbildung 1 (siehe Anhang) lässt sich ein POSIX Prozess wie folgt beschreiben: Wenn sequentiell ein Prozess A gestartet wird, so wird aus diesem Prozess A ein parent Prozess, sobald der Befehl fork() ausgeführt wird. Nach diesem Befehl werden die aktuell

---

[5] Vgl. IEEE & The Open Group: A Backgrounder on IEEE Std 1003.1, Homepage: http://www.opengroup.org/austin/papers/backgrounder.html, Letzter Aufruf: 14.01.19

[6] Vgl. Stroup, Loren: POSIX Threads, Presentation: http://www.cs.ucf.edu/~lboloni/Teaching/EEL6897_2007/presentations/Loren-Stroup-POSIXThreads.ppt, Letzter Aufruf: 14.01.19

[7] Vgl. The Open Group: pthread.h, Publikation: http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html#tag_13_35, Letzter Aufruf: 14.01.19

en Informationen des parents A in ein child Prozess kopiert. Mit dem Befehl exec() lässt sich auf Basis des child Prozesses ein neuer Prozess starten. Der parent Prozess kann, sofern vorgesehen, parallel weitere Befehle ausführen. Mit dem child Prozess lassen sich weitere Prozesse aufrufen. Mit dem Befehl exit() wird wieder zum parent Prozess zurückgelangt. Mit dem Befehl kill() lässt sich ein Prozess beenden. Innerhalb dieses Prozesses können Signale zwischen dem parent und dem child Prozess übergeben werden.[8]
Diese Befehle sieht die POSIX library vor, wodurch dieser POSIX Prozess definiert ist.

Neben den pthreads und dem POSIX Prozess gibt es noch den Dateizugriff unter POSIX beziehungsweise die POSIX I/O, welches eine zentrale Funktion bezüglich Datei- und Speichersysteme einnimmt. Der Standard definiert Funktionen wie fopen() zum Öffnen von Dateien, fwrite() zum Schreiben in Dateien, fread() zum Lesen in Dateien und fclose() zum Schließen von Dateien. Diese Befehle werden im POSIX Standard mit Anforderungen unterlegt, so dass beispielsweise der Befehl fwrite() nach dem Schreiben unter anderem nochmal überprüfen soll, ob auch wirklich in die Datei geschrieben wurde.

Neben der POSIX I/O gibt es auch noch den Terminal I/O, beziehungsweise die POSIX Datenströme. POSIX definiert Standards in Bezug auf die Ein- und Ausgabe auf dem Terminal. stdin ist die Standardeingabe unter POSIX, mit welcher Daten in ein Programm eingelesen werden können. Anwendung für diesen Fall ist die Nutzung des Keyboards, wobei die Signale per stdin eingegeben werden. Außerdem gibt es noch stdout, um Daten aus dem Programm auf dem Display auszugeben, oder stderr, um Fehler- und Statusmeldungen auszugeben.[9] POSIX I/O gerät zunehmend unter Kritik. Damit wird sich im Punkt 3.3 erneut befasst.

Neben den von POSIX vorgesehenen Befehlen gibt es noch POSIX Ausdrücke. Diese Ausdrücke muss ein Betriebssystem annehmen,

[8] Vgl. Abbildung 1 im Anhang
[9] Vgl. Abbildung 2 im Anhang

damit dieser als POSIX konform bezeichnet werden kann. In ASCII gibt es beispielsweise einen Ausdruck für alle Alphanumerischen Zeichen: [a-zA-Z0-9]. Dieser Ausdruck als POSIX Ausdruck lautet: [:alnum:]. Ein weiteres Beispiel ist der ASCII-Ausdruck [A-Z] für alle großgeschriebenen Buchstaben. Der POSIX Ausdruck dazu lautet [:upper:]. [10]

Diese Beispielauszüge und –inhalte der POSIX Spezifikation zeigen eine Auswahl der Eigenschaften des POSIX-Standards. Um einen Überblick über die verschiedenen Versionen und Features des Standards zu erhalten, sollen im Folgenden die Versionen der POSIX Spezifikation genannt und bezüglich der jeweiligen Inhalte erläutert werden.

3.1.2 POSIX – Versionsverlauf

POSIX.1: IEEE Std 1003.1-1988. Dies gilt als erster POSIX-Standard. In dieser ersten Version wurden Core Services durch die POSIX Spezifikation definiert, wie Prozesserstellung und –kontrolle, C Bibliothek, Speicher- und Dateisystem Operationen, Signale (Exceptions, Fehlerhafte Aufrufe, etc.).

Nach diesem Standard folgte der Standard POSIX.1b: IEEE Std 1003.1b-1993. In diesem Update wurden Real-time extensions hinzugefügt. Diese Real-time Funktionen wurden in dieser Version des Standards unter anderem durch die Definition von geteilten Speichern, Timer und Real-time Signalen realisiert. Diese Real-time extensions werden unter anderem für Video- und Audiostreaming benötigt. 11

Anschließend folgt die Version POSIX.1c: IEEE Std 1003.1c-1995. In dieser Version wurden die Pthreads veröffentlicht, welche unter

[10] Goyvaerts, Jan: POSIX Bracket Expressions, Web: https://www.regular-expressions.info/posixbrackets.html, Letzter Aufruf: 14.01.2019
[11] Obenland, Kevin M.: POSIX in Real-Time, Artikel: https://www.eetimes.com/document.asp?doc_id=1200444 Letzter Aufruf: 14.01.2019

Punkt 3.1.1.2 behandelt wurden. Diese Version des POSIX Standards hat beispielsweise die Thread Creation, oder das Thread Scheduling ergänzt, mit welchen die Pthreads erstellt werden können.

POSIX.2: IEEE Std 1003.2-1992. Dies ist die vierte Version, in welcher Shell Utilities von dem IEEE definiert wurden. In dieser Version wurden Hilfsprogramme dargestellt und die Standardschnittstelle auf Quellcodeebene definiert.

1997 hat sich die Austin Group geformt. Die anschließend definierten Standards fassen vorherige Standards in einer Spezifikation zusammen. So erschien 2001 die Spezifikation POSIX.1-2001. Diese hat dieselbe Struktur wie der heutige POSIX Standard: Basis Definitionen, ein System Interface und die Header und die Befehle und Hilfsprogramme.

Mit POSIX.1-2004 wurde ein kleines Update veröffentlicht, dass die neuen technischen Anforderungen abbilden solle.

2008 wurde der Standard POSIX.1-2008 herausgegeben. Dieser beinhaltet ein größeres Update als die Vorgängerversionen. Es wurden in diesem Standard neue technische Anforderungen berücksichtigt und dementsprechend neue Basis-Definitionen, etc. aufgestellt. Dieser Standard gilt auch heute noch als POSIX-Standard und trägt den offiziellen Namen ISO/IEC/IEEE 9945.

Nach 2008 wurden diverse kleinere Updates herausgegeben. 2017 beispielsweise wurde der Standard POSIX.1-2017 herausgegeben, welcher technische Änderungen berücksichtigten soll.

3.2 POSIX – Verwendung & Relevanz

Diverse Unix und Unix-Like Systeme sind POSIX konform. Darunter A/UX, ein Unix Betriebssystem der Firma Apple, Linux, oder HP-UX, ein Unix Betriebssystem der Firma Hewlett Packard. Neben diesen

sind auch andere nicht UNIX Betriebssysteme teilweise, oder komplett POSIX konform, wie beispielsweise das Betriebssystem das Windows NT, welches für Workstations verwendet wurde.[12]
Die Einbeziehung der POSIX Spezifikation in die Entwicklung der meisten großen Betriebssysteme hat unter anderem den Grund, dass der Standard grundlegende Funktionen anbietet, die im Kontext der Programmierung als trivial angesehen werden. Ein Beispiel dieser Eigenschaft sind die Befehle open(), write(), read(), etc.[13]

Heute (Stand: 2018) gibt es drei große Mobil- und Desktop-orientierte Unix Systeme: MAC OS X, Android und Ubuntu, welche alle POSIX konform, beziehungsweise POSIX kompatibel sind.
Dies deutet daraufhin, dass POSIX eine hohe Reichweite hat.
Es ist damit auch der größte Standard für Betriebssysteme auf dem Markt, mit einer verbreiteten Verwendung bei der Entwicklung von Betriebssystemen, was diesem Standard eine hohe Relevanz zukommen lässt.[14]

3.3 POSIX – Kritische Betrachtung

Wie unter dem Punkt 3.1.1.1 Spezifikation aufgezeigt, galt bei der erstmaligen Erstellung des POSIX Standards das Prinzip, dass minimale Änderung an bestehender Anwendungssoftware notwendig sein sollten, sobald der Standard fertig definiert wurde. Neben weiteren Faktoren führe dies dazu, dass die POSIX Spezifikation kritisiert wird, da diese nicht den modernen technischen Anforderungen angepasst sei.[15]

[12] Wikipedia: POSIX, Web: https://en.wikipedia.org/wiki/POSIX#cite_note-16, Letzter Aufruf: 14.01.2019
[13] Lockwood, Glenn: What´s so bad about POSIX I/O?, Publikation: https://www.nextplatform.com/2017/09/11/whats-bad-posix-io/, Letzter Aufruf: 14.01.2019
[14] Columbia Univ.: POSIX Has Become Outdated, Publikation: http://www.cs.columbia.edu/~vatlidak/resources/POSIXmagazine.pdf, Letzter Aufruf: 14.01.2019
[15] Columbia Univ.; Lockwood, Glenn: Quelle wie angegeben: Fußnoten 13, 14

In diesem Abschnitt sollen zwei Hauptargumente der Kritiker präsentiert werden, damit der POSIX Standard aus einer kritischen Perspektive betrachtet werden kann.

Ein Argument der Kritiker ist die Tatsache, dass der POSIX Standard den Aufbau von Metadaten fest vorschreibt. Dies führe dazu, dass dieses Vorschreiben zwar zur Vereinheitlichung führe, dennoch als zu vorschreibend, beziehungsweise als inflexibel wahrgenommen wird. POSIX sieht beispielsweise vor, dass unter anderem der Urheber der Datei, das Erstellungsdatum, oder das Datum der letzten Bearbeitung als Metadaten gepflegt werden. Dies wird insofern kritisiert, als dass es laut Kritiker in manchen Fällen als nicht notwendig betrachtet wird, stets den User anzugeben, wenn man sich beispielsweise in demselben Directory befindet. In anderen Fällen könne man laut Kritiker mehr Metadaten benötigen, als von POSIX vorgesehen. So würden README Dateien erstellt werden, welche eigentlich als Metadaten vorgesehen sein sollten.[16]

Ein weiteres Argument resultiert aus der Tatsache, dass POSIX zustandsorientiert ist. Wenn unter POSIX I/O Daten geschrieben und gelesen werden, so geschieht dies über einen persistenten Zustand, indem durch das Betriebssystem mit Hilfe von Datei-Handle die Aktion vollzogen wird. Das Betriebssystem muss bei jedem Datei-Handle, also bei jedem Schreib-, oder Leseprozess, den Zustand des Datei-Handles abfragen, was laut Kritiker zu einem Hindernis in Bezug auf die Skalierbarkeit von Dateizugriffen führt, welche in heutigen Anwendungen in Millionen und Milliarden vorkommen. Im Kontext des High Performance Computings soll unter anderem diese Eigenschaft dazu führen, dass POSIX Systeme diesbezüglich eine Barriere darstellen.[17]

4 Fazit

Der POSIX Standard wurde von dem IEEE entwickelt und hatte die Zielsetzung die Portabilität von Anwendungssoftware zu sichern. Die

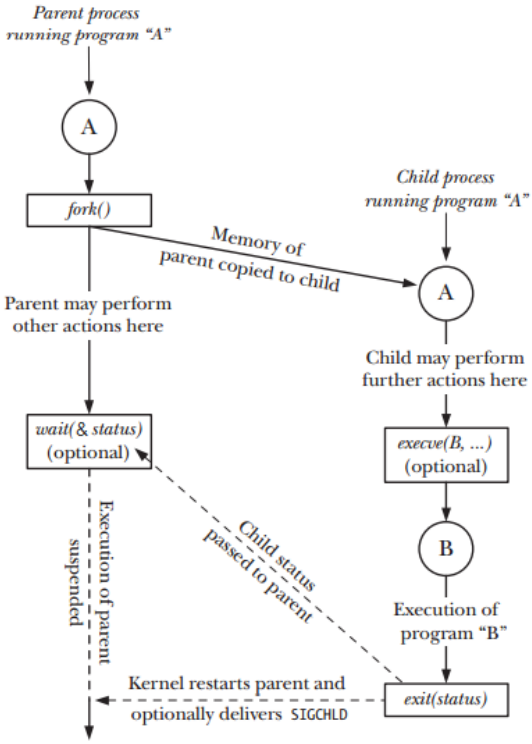
[16] Lockwood, Glenn: Quelle wie angegeben: Fußnote 13
[17] Lockwood, Glenn: Quelle wie angegeben: Fußnote 13

Zielerreichung dieses Ziels lässt sich einerseits durch die weite Verwendung des Standards bestätigen. Andererseits können die Anwendungssoftwarehersteller sich durch die POSIX Spezifikation um die Funktionalität der Software kümmern, ohne spezielle Implementationen vorzunehmen, um die Portabilität zu gewährleisten, was ebenfalls mit der Zielsetzung von POSIX einhergeht. Es werden auch heute noch Elemente aus POSIX, beispielsweise die POSIX Threads, von den meisten Betriebssystemen unterstützt und eingesetzt.

Es wurden jedoch nach 2008 keine großen Änderungen an dem Standard vorgenommen, so dass die Kritik an der Spezifikation steigt und POSIX als veraltet angesehen wird. Vorschläge für eine Verbesserung des Standards werden zum aktuellen Zeitpunkt bereits erarbeitet, was zu einem neuen großen Update des POSIX Standards führen könnte. Weitere Entwicklungen und Bemühungen werden von verschiedenen Organisationen, Institutionen und Unternehmen vorangetrieben, was eventuell zu einem neuen Standard führen kann.

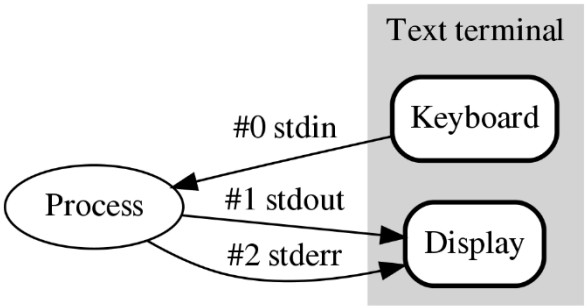
Anhang

Abbildung 1:



Quelle: Übernommen aus: „Linux Processes and Signals“, https://www.bo-gotobogo.com/Linux/linux_process_and_signals.php

Abbildung 2:



Quelle: Übernommen aus: „Standard-Datenströme“, https://de.wikipedia.org/wiki/Standard-Datenstr%C3%B6me, 2018

Quellenverzeichnis

Atlidakis, Vaggelis; Andrus, Jeremy; Geambasu, Roxana; Mitropoulos, Dimitris; Nieh, Jason: POSIX Has Become Outdated, http://www.cs.columbia.edu/~vatlidak/resources/POSIXmagazine.pdf, 2016

Emmerson, Steve: LDM & POSIX threads, https://www.unidata.ucar.edu/mailing_lists/archives/ldm-users/2002/msg00512.html, 2002

Goyvaerts, Jan: POSIX Bracket Expressions, https://www.regular-expressions.info/posixbrackets.html, 2019

IEEE: Website, https://www.ieee.org, 2019

IEEE; The Open Group: The Open Group Base Specifications Issue 7, http://pubs.opengroup.org/onlinepubs/9699919799.2016edition/, 2016

IEEE; The Open Group: The Open Group Base Specifications Issue 7, http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html#tag_13_35, 2018

Indiana University: About POSIX – Knowledge Base, https://kb.iu.edu/d/agjv, 2019

Johnson, Ross: Pthreads Win32, https://sourceware.org/pthreads-win32/, 2012

Lockwood, Glenn: What´s so bad about POSIX I/O? , https://www.nextplatform.com/2017/09/11/whats-bad-posix-io/, 2017

N.N.: IEEE Posix Standards, https://de.slideshare.net/djhseen1993/posix-46606116, 2015

Obenland, Kevin M.: POSIX in Real-Time, https://www.eetimes.com/document.asp?doc_id=1200444, 2001

Simonsen, Keld: JTC1/SC22/WG15 – POSIX! , http://www.open-std.org/jtc1/sc22/WG15/, 2019

Stroup, Loren: POSIX Threads, http://www.cs.ucf.edu/~lboloni/Teaching/EEL6897_2007/presentations/LorenStroup-POSIXThreads.ppt, 2014

The Open Group: A Backgrounder on IEEE Std 1003.1, http://www.opengroup.org/austin/papers/backgrounder.html, 2019

The Open Group: Website, https://www.opengroup.org/, 2019

Wikipedia: Austin Group, https://en.wikipedia.org/wiki/Austin_Group, 2019

Wikipedia: C POSIX library, https://en.wikipedia.org/wiki/C_POSIX_library, 2018

Wikipedia: Native POSIX Thread Library, https://de.wikipedia.org/wiki/Native_POSIX_Thread_Library, 2015

Wikipedia: Open Group, https://de.wikipedia.org/wiki/Open_Group, 2016

Wikipedia: Portable Operating System Interface, https://de.wikipedia.org/wiki/Portable_Operating_System_Interface, 2018

Wikipedia: POSIX, https://en.wikipedia.org/wiki/POSIX#cite_note-16, 2018

Wikipedia: Standard-Datenströme, https://de.wikipedia.org/wiki/Standard-Datenstr%C3%B6me, 2018

Videos:

Morse, Shannon; Hak5: What is POSIX in Unix? Linux Terminal 201 - HakTip 161, https://www.youtube.com/watch?v=U0GbJtnfqSM, 2017

Elliot, Tom; Chuck Severance: Understanding the POSIX Open System Reference Model (1990), https://www.youtube.com/watch?v=31bS6cUHj-U, 2007

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

……

Unterschrift des Verfassers