Moduldokumentation

Modul Systemprogrammierung (syspr)

Simon Wächter

2017

Inhalt

[1 Einleitung 4](#_Toc478504026)

[1.1 Einleitung 4](#_Toc478504027)

[1.2 Lernziele 4](#_Toc478504028)

[1.3 Prüfungen 4](#_Toc478504029)

[2 Woche 1 5](#_Toc478504030)

[2.1 TIOBE Index 5](#_Toc478504031)

[2.2 Grundzüge der Programmierung in einer prozeduralen Sprache 5](#_Toc478504032)

[2.3 Kontrollflüsse und Schleifen 6](#_Toc478504033)

[2.4 Zusammengesetzte Datentypen und Zeiger 6](#_Toc478504034)

[2.5 Aufbau einer Programmierumgebung 6](#_Toc478504035)

[2.6 Gründe für C 7](#_Toc478504036)

[2.7 Dateien in C 7](#_Toc478504037)

[2.8 Das erste C Programm 7](#_Toc478504038)

[2.9 Ein komplexeres C Programm 8](#_Toc478504039)

[2.10 ASCII Tabelle 8](#_Toc478504040)

[3 Woche 2 9](#_Toc478504041)

[3.1 Präprozessor 9](#_Toc478504042)

[3.2 Symbolische Konstanten, Namen und Markos 9](#_Toc478504043)

[3.3 Header Datei als Beispiel 9](#_Toc478504044)

[3.4 Operationen break und continue 9](#_Toc478504045)

[3.5 Übungsbeispiel 10](#_Toc478504046)

[3.6 Operation switch 10](#_Toc478504047)

[3.7 Datentyp struct 11](#_Toc478504048)

[3.8 Funktionsaufrufe 11](#_Toc478504049)

[3.9 Pointer auf ein Objekt im Speicher 12](#_Toc478504050)

[3.10 Übung 12](#_Toc478504051)

[3.11 Fehlersuche und Fehlerbehebung 13](#_Toc478504052)

[4 Woche 3 14](#_Toc478504053)

[5 Woche 4 15](#_Toc478504054)

[5.1 Einführung in die Win32 API 15](#_Toc478504055)

[5.2 Bemerkung zu Standard beim Programmieren 15](#_Toc478504056)

[5.3 Win32 API und C Syntax 15](#_Toc478504057)

[5.4 WinMain als Einstiegspunkt für Windows-Programme 16](#_Toc478504058)

[5.5 Win32 Konsolenprogramm mit Visual Studio 16](#_Toc478504059)

[5.6 Fehlermeldungen mit FormatMessage und GetLastError 17](#_Toc478504060)

[5.7 Win32 Applikation mit Visual Studio 18](#_Toc478504061)

[5.8 Prozesszeiten 19](#_Toc478504062)

[6 Woche 5 22](#_Toc478504063)

[6.1 Kontext von Dateisystemen 22](#_Toc478504064)

[6.2 Anforderungen an ein Dateisystem 22](#_Toc478504065)

[6.3 Aufbau eines Unix Dateisystems 22](#_Toc478504066)

[6.4 Beispiel Linux Dateisystem ext2 23](#_Toc478504067)

[6.5 Gliederung von ext2 23](#_Toc478504068)

[6.6 Dateien unter Linux 23](#_Toc478504069)

[6.7 Index Nodes als sogenannte Inodes 24](#_Toc478504070)

[6.8 Design und Aufbau einer Inode 24](#_Toc478504071)

[6.9 Verzeichnisse 24](#_Toc478504072)

[6.10 Hard- und Symbolic Links 25](#_Toc478504073)

[6.11 Virtuelles Dateisystem 25](#_Toc478504074)

[6.12 VFS Objekte und Datenstrukturen 25](#_Toc478504075)

[6.13 Superblock 26](#_Toc478504076)

[6.14 Operationen auf Superblocks 26](#_Toc478504077)

[6.15 Inodes 26](#_Toc478504078)

[6.16 Struktur struct inode 27](#_Toc478504079)

[6.17 Operationen auf inode 27](#_Toc478504080)

[6.18 Struktur struct dentry 27](#_Toc478504081)

[6.19 Struktur struct file 28](#_Toc478504082)

[6.20 Operationen auf file 28](#_Toc478504083)

[6.21 Grundlegende Dateioperationen 28](#_Toc478504084)

[6.22 System Calls im Linux Dateisystem 29](#_Toc478504085)

[6.23 Library Calls zur Verzeichnismanipulation 29](#_Toc478504086)

[6.24 Beispiel einer Verzeichnismanipulation 29](#_Toc478504087)

[6.25 System Calls zur Dateimanipulation 30](#_Toc478504088)

[6.26 Beispiel einer Dateimanipulation 31](#_Toc478504089)

[6.27 System Calls zur Arbeit im Dateisystembaum 33](#_Toc478504090)

[6.28 Grundlegende API Funktionen in Windows 33](#_Toc478504091)

[7 Woche 6 35](#_Toc478504092)

[7.1 Einführung in Prozesse 35](#_Toc478504093)

[7.2 Unterschied zwischen Prozesstask und Programm 35](#_Toc478504094)

[7.3 Lebenszyklus eines Prozesses 35](#_Toc478504095)

[7.4 Verwaltung von Prozessen 36](#_Toc478504096)

[7.5 Anforderungen an ein Prozesssteuersystem 36](#_Toc478504097)

[7.6 User- und Kernel Mode 37](#_Toc478504098)

[7.7 Prozess-/Kontextwechsel 37](#_Toc478504099)

[7.8 Prozesskontext 37](#_Toc478504100)

[7.9 Scheduler 38](#_Toc478504101)

[7.10 Schedulingstrategien 38](#_Toc478504102)

[7.11 Prozessbezogene System Calls in Unix 38](#_Toc478504103)

[7.12 System Call exec() 39](#_Toc478504104)

[7.13 System Call fork() 39](#_Toc478504105)

[7.14 Vergleich fork() und exec() 40](#_Toc478504106)

[7.15 System Call exit() 40](#_Toc478504107)

[7.16 Funktionen atexit() und on\_exit() 40](#_Toc478504108)

[7.17 System Call wait() 40](#_Toc478504109)

[7.18 Programmierung mit wait() 41](#_Toc478504110)

[7.19 Auswertung des Exit Code nach wait() 41](#_Toc478504111)

[7.20 Prozesse erzeugen mittels Windows API 42](#_Toc478504112)

[7.21 Beispiel von kill in der Windows API 42](#_Toc478504113)

[7.22 Beispiel der Dateisystemüberwachung in der Windows API 43](#_Toc478504114)

[8 Woche 7 44](#_Toc478504115)

# Einleitung

## Einleitung

Dieses Dokument stellt die Moduldokumentation für das Modul syspr dar. Allfällige Unterlagen sind im Modulordner zu finden.

## Lernziele

Das Modul beinhaltet folgende Lernziele:

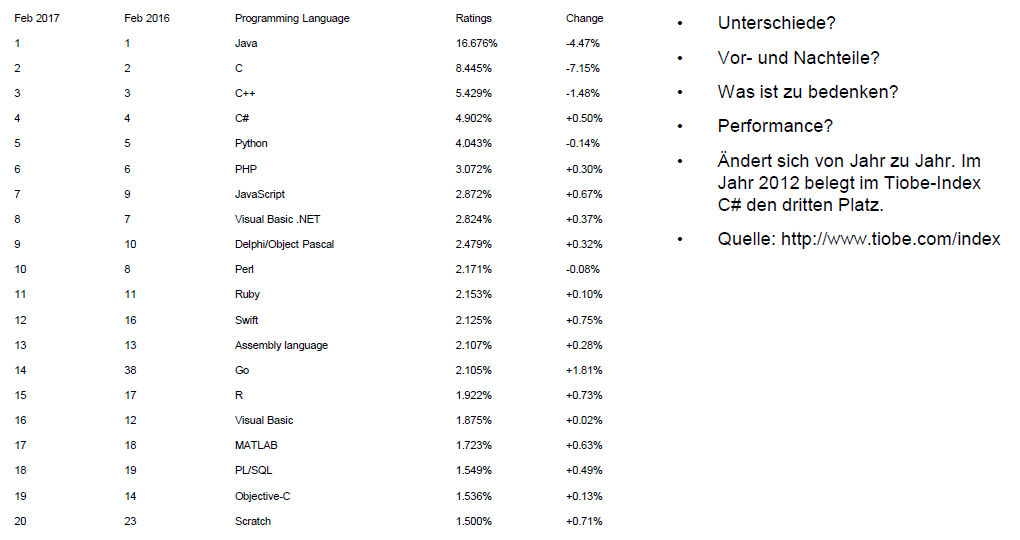
* Kennenlernen des Betriebssystems, beziehungsweise der wesentlichen Systemressourcen aus Sicht des Programmierers:
  + Dateisystem
  + Prozesssteuerung
  + Interprozesskommunikation
  + Synchronisation
* Kennenlernen der Windows API
* Zusammenhang der Windows API mit anderen Programmiersprachen
* Kennenlernen der Programmiersprache C

## Prüfungen

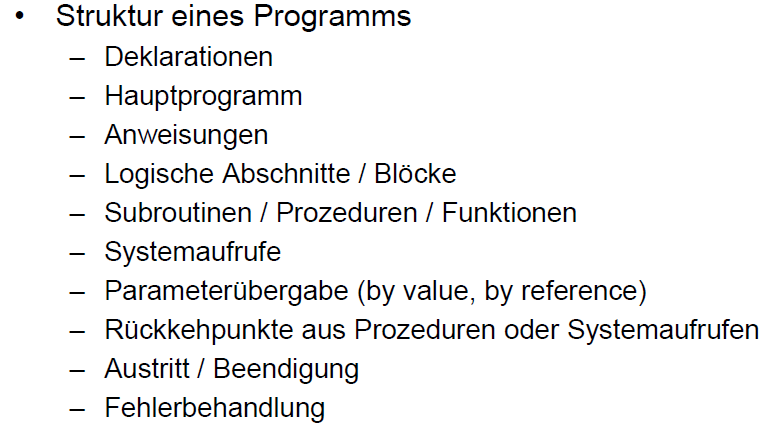
Die Modulnote setzt sich zu 100% aus zwei Semesterprüfungen zu je 50% zusammen.

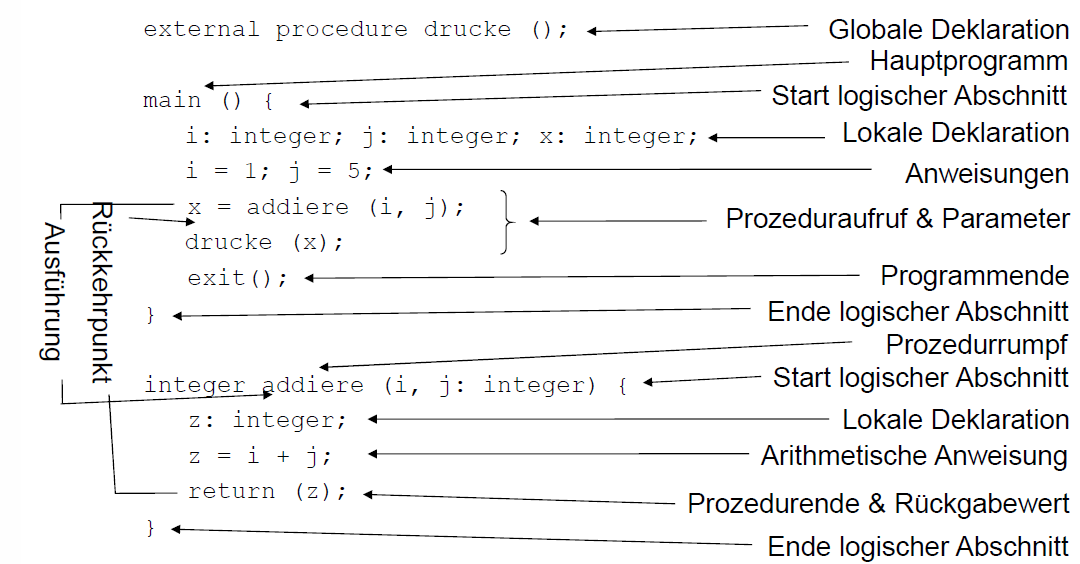
# Woche 1

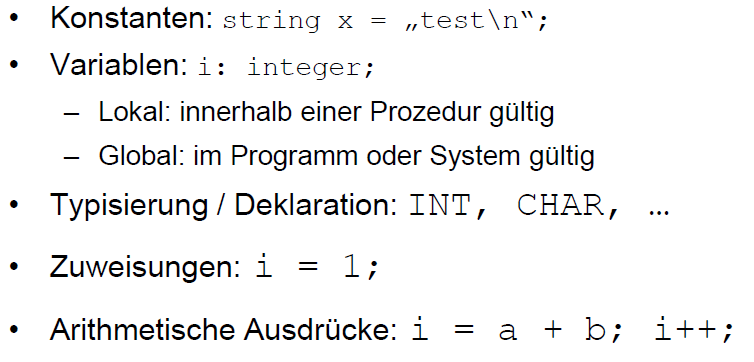
## TIOBE Index



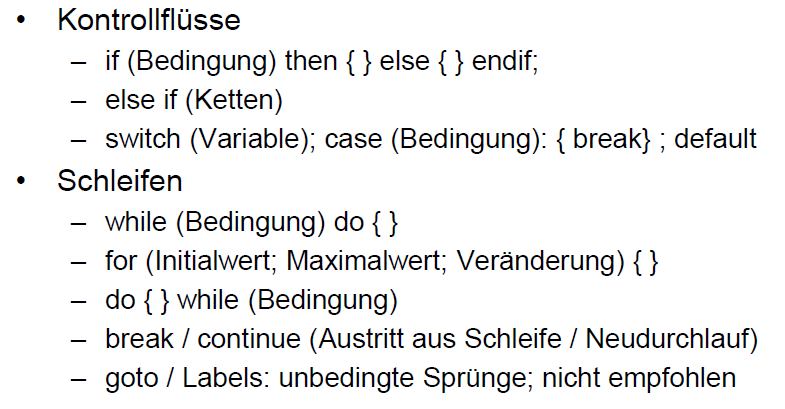
## Grundzüge der Programmierung in einer prozeduralen Sprache



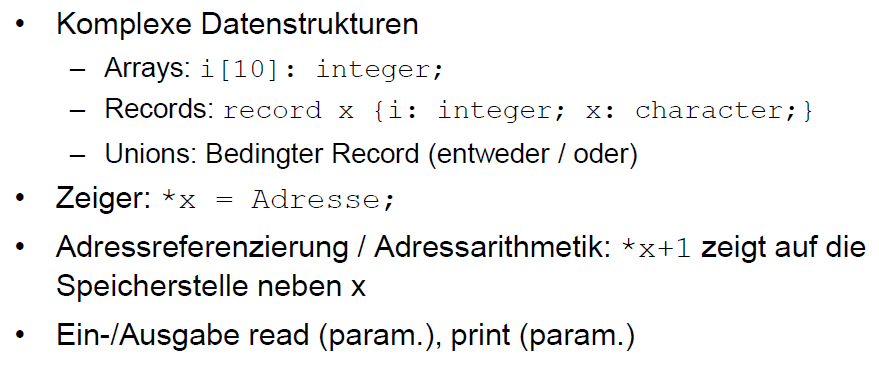




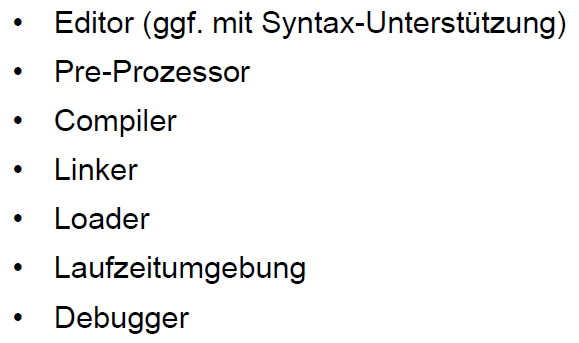
## Kontrollflüsse und Schleifen



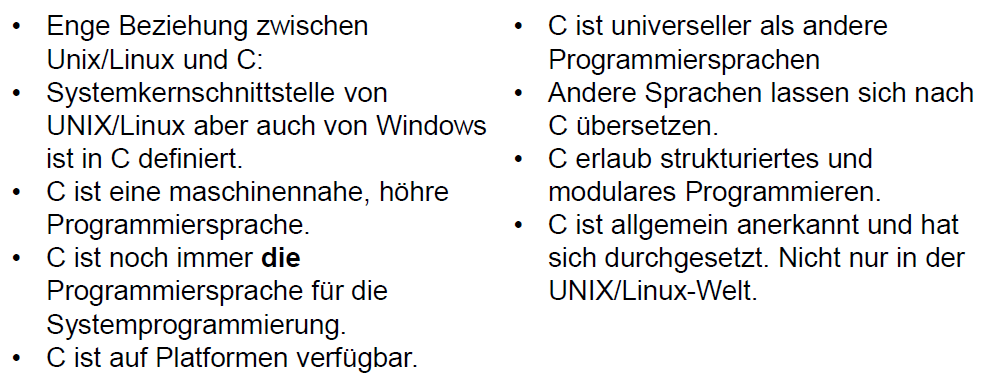
## Zusammengesetzte Datentypen und Zeiger



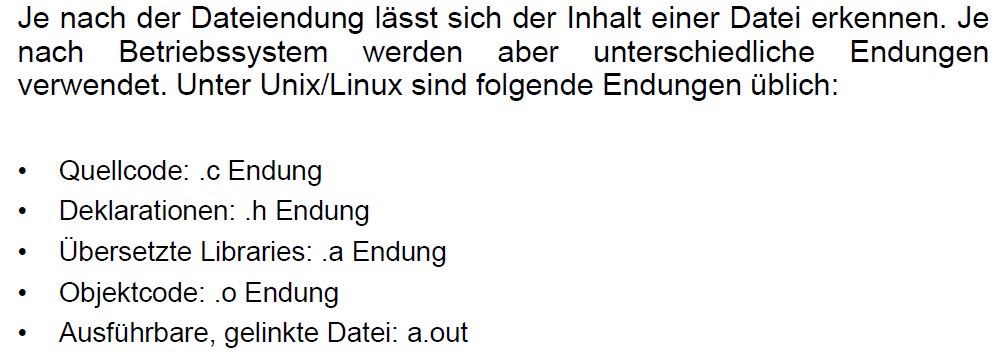
## Aufbau einer Programmierumgebung



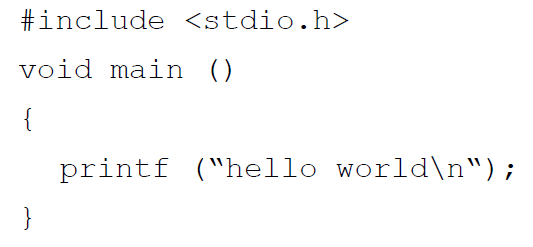
## Gründe für C



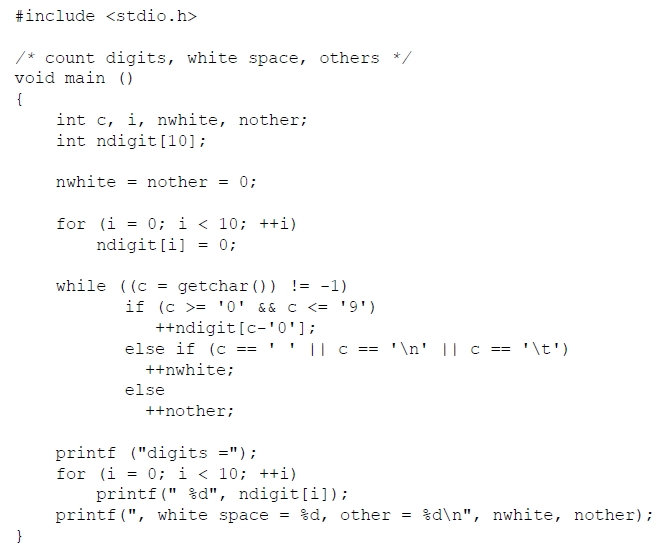
## Dateien in C



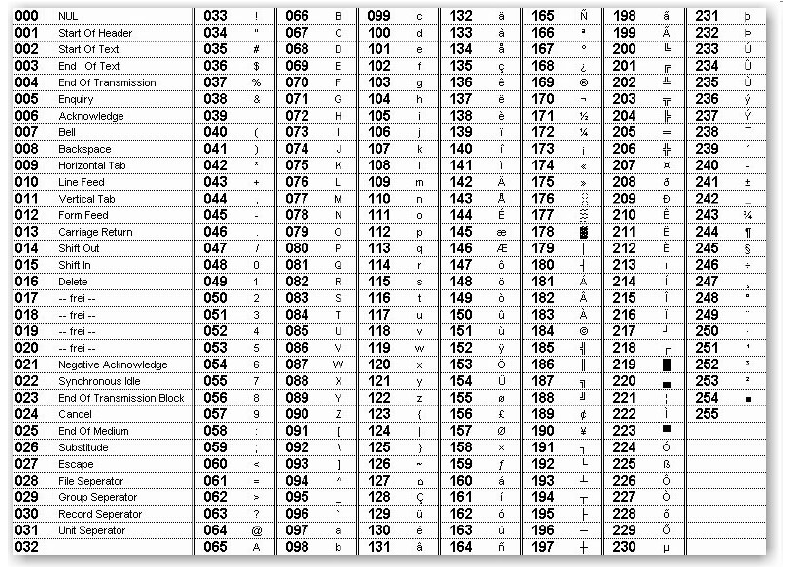
## Das erste C Programm



## Ein komplexeres C Programm

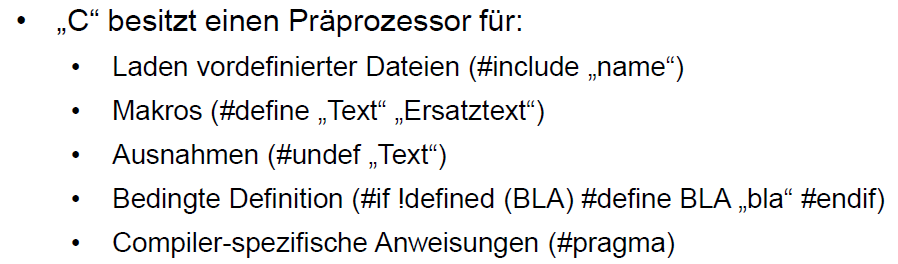


## ASCII Tabelle



# Woche 2

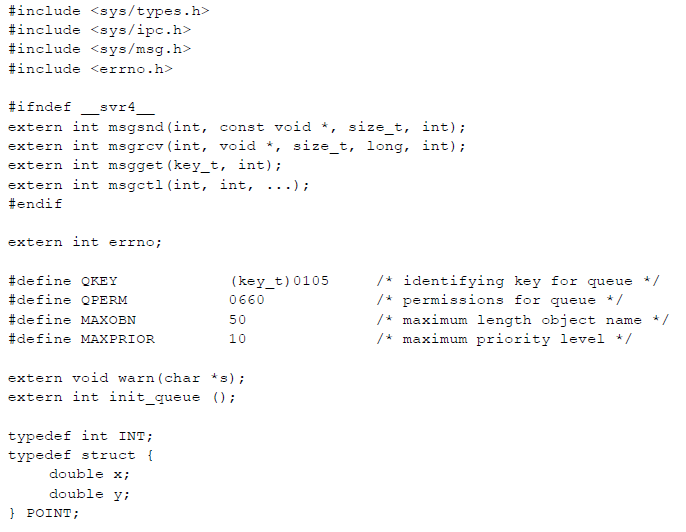
## Präprozessor



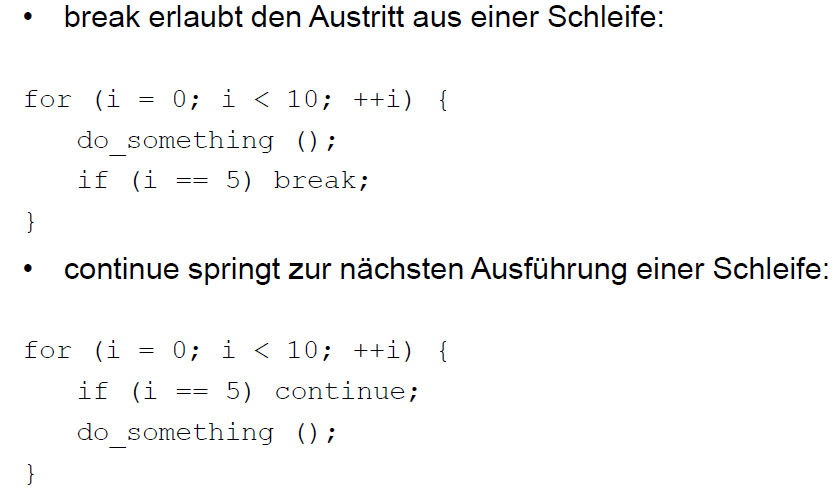
## Symbolische Konstanten, Namen und Markos



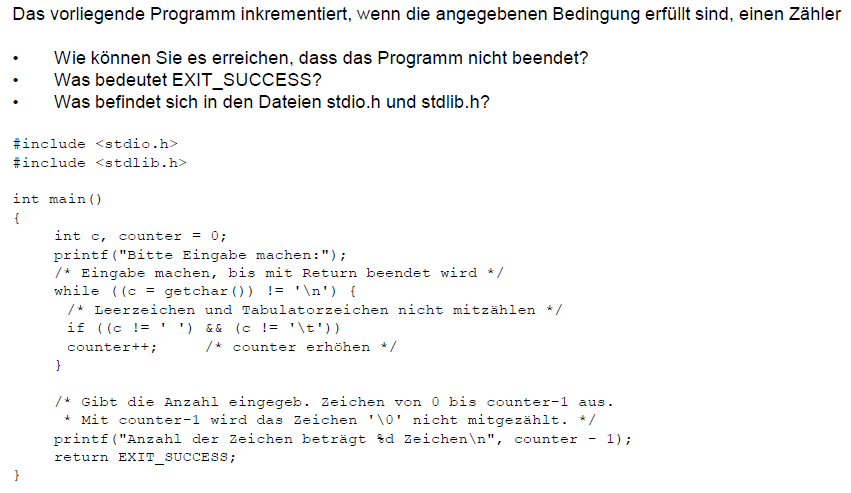
## Header Datei als Beispiel

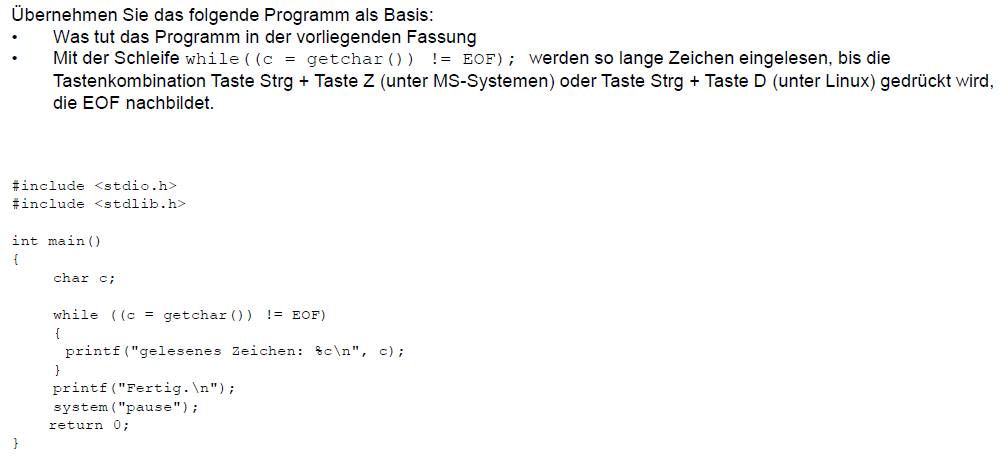


## Operationen break und continue

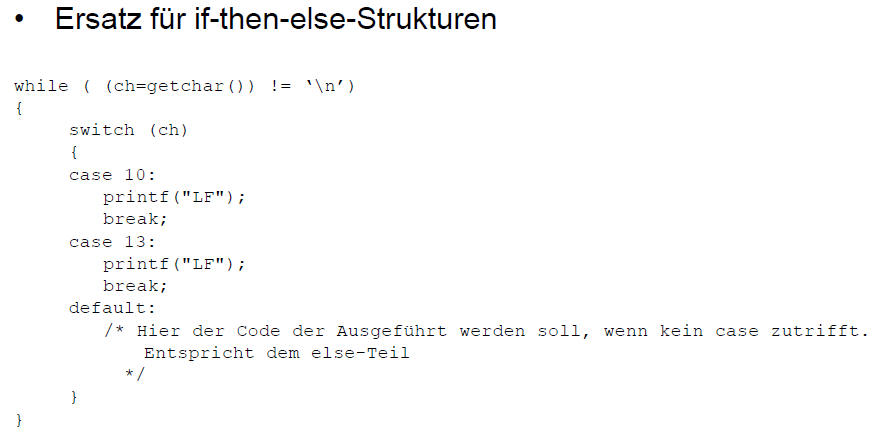


## Übungsbeispiel

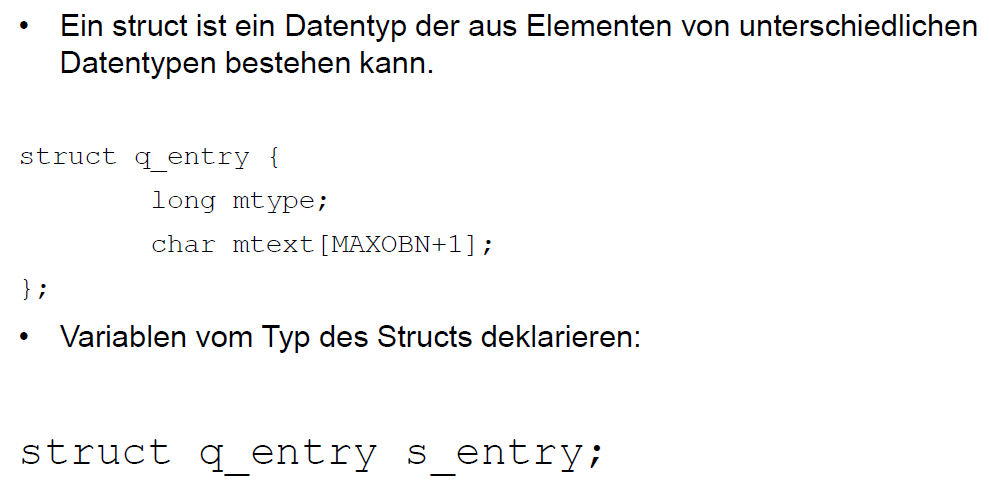


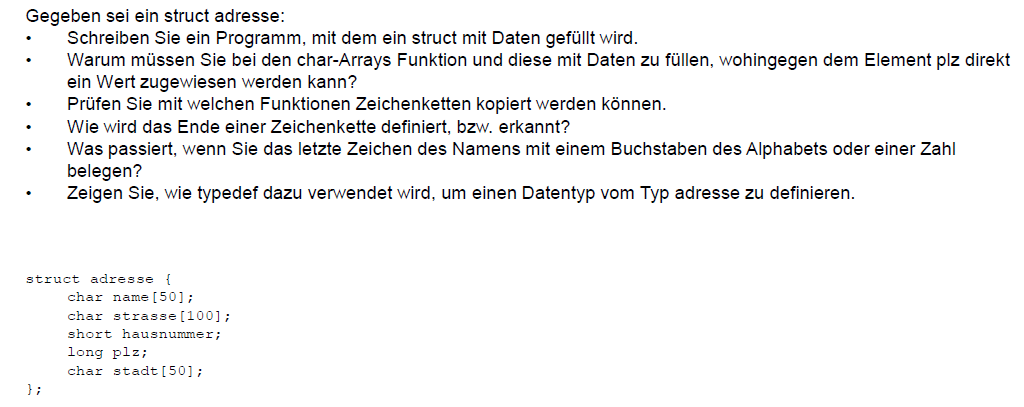


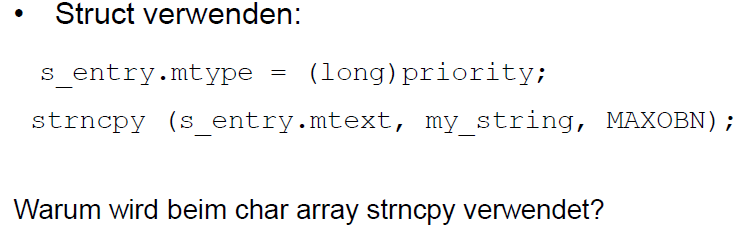
## Operation switch



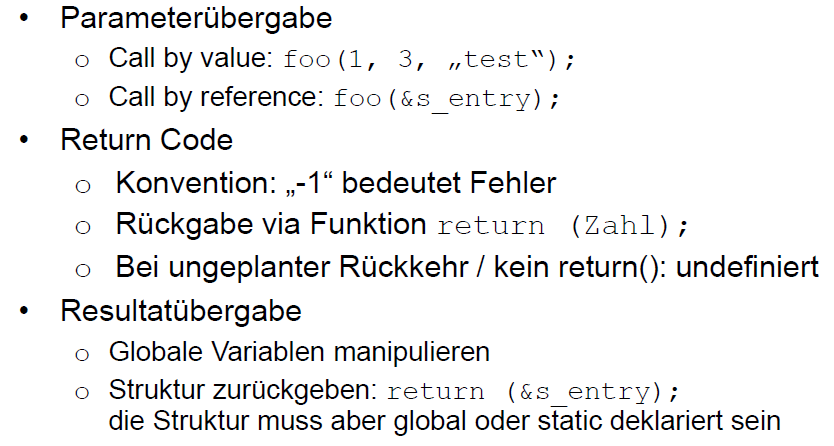
## Datentyp struct



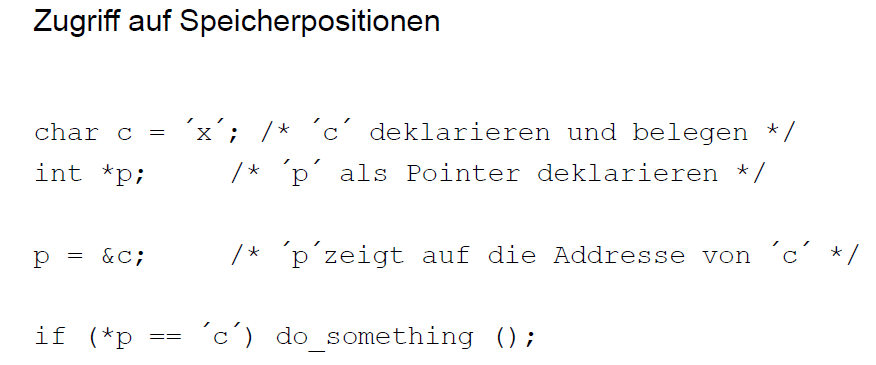


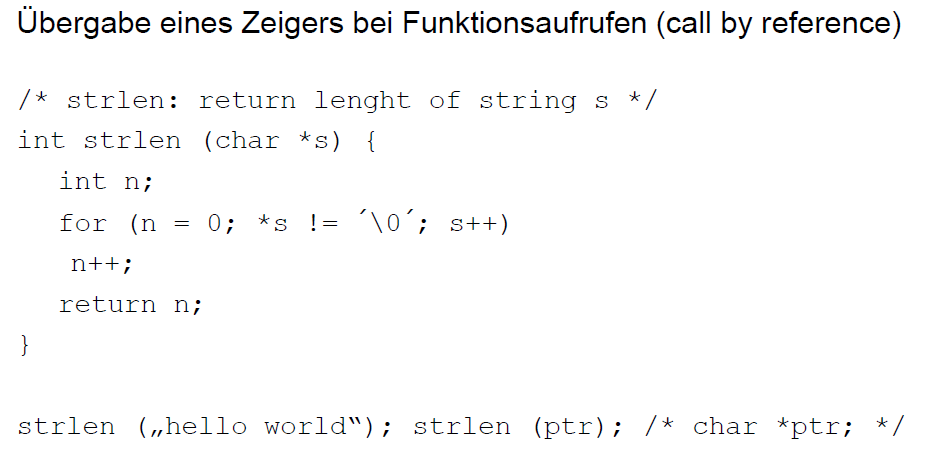


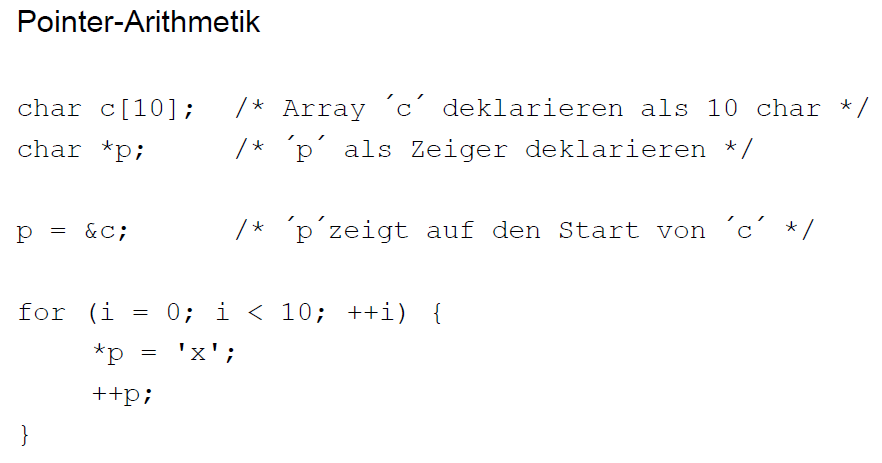
## Funktionsaufrufe



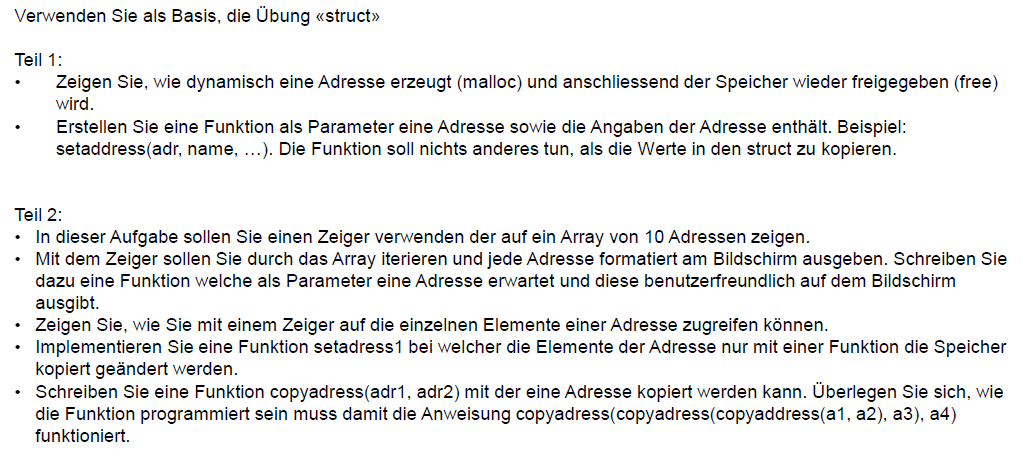
## Pointer auf ein Objekt im Speicher



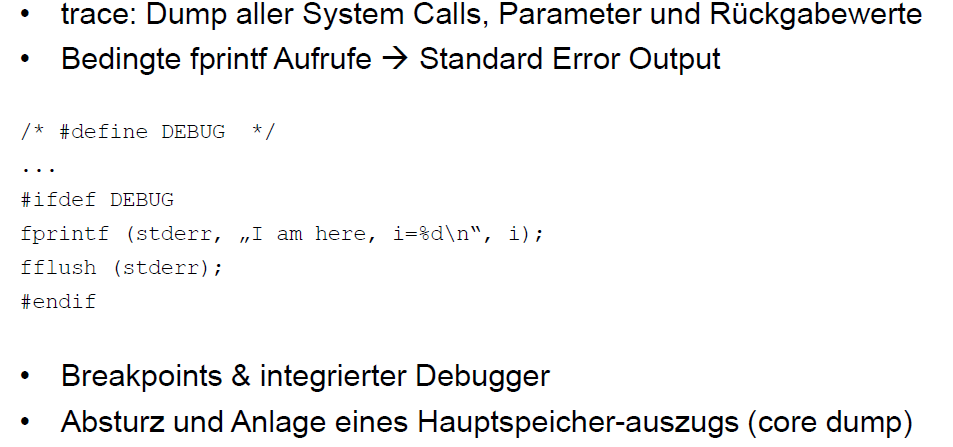




## Übung



## Fehlersuche und Fehlerbehebung

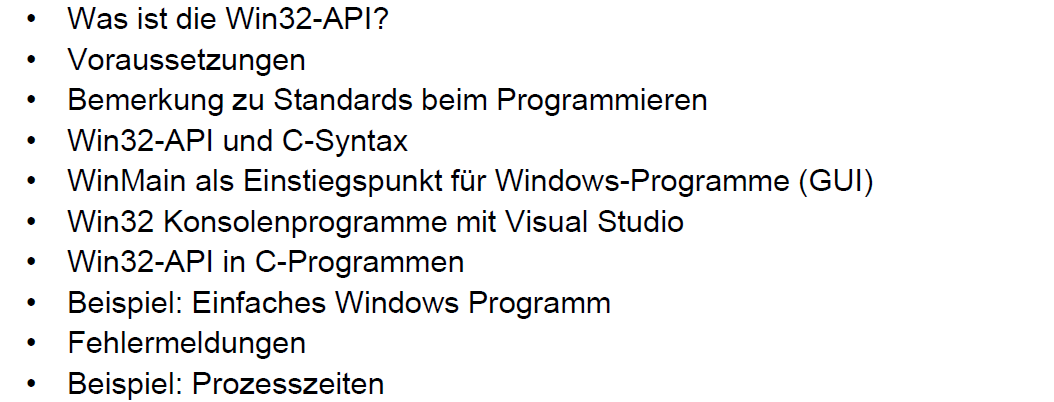


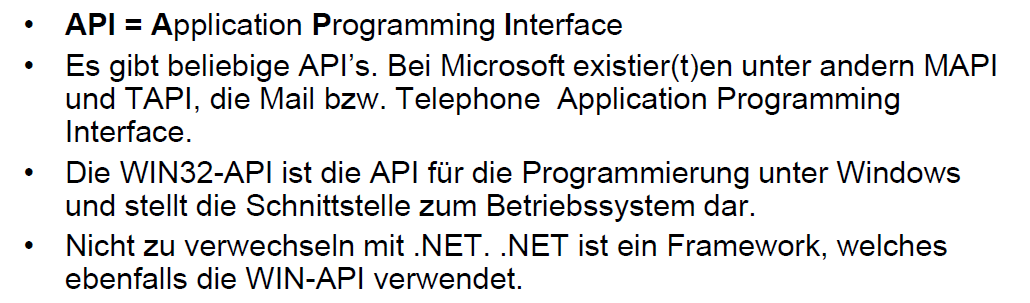
# Woche 3

In Woche 3 fiel krankheitsbedingt der Unterricht aus.

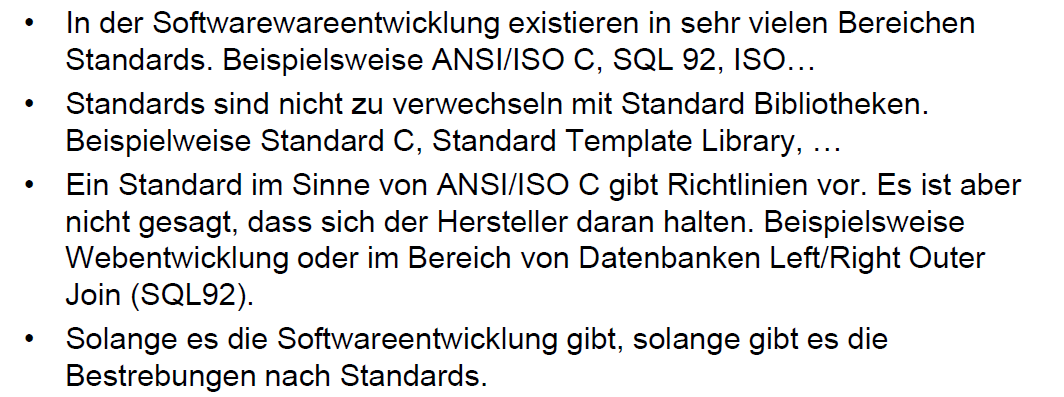
# Woche 4

## Einführung in die Win32 API

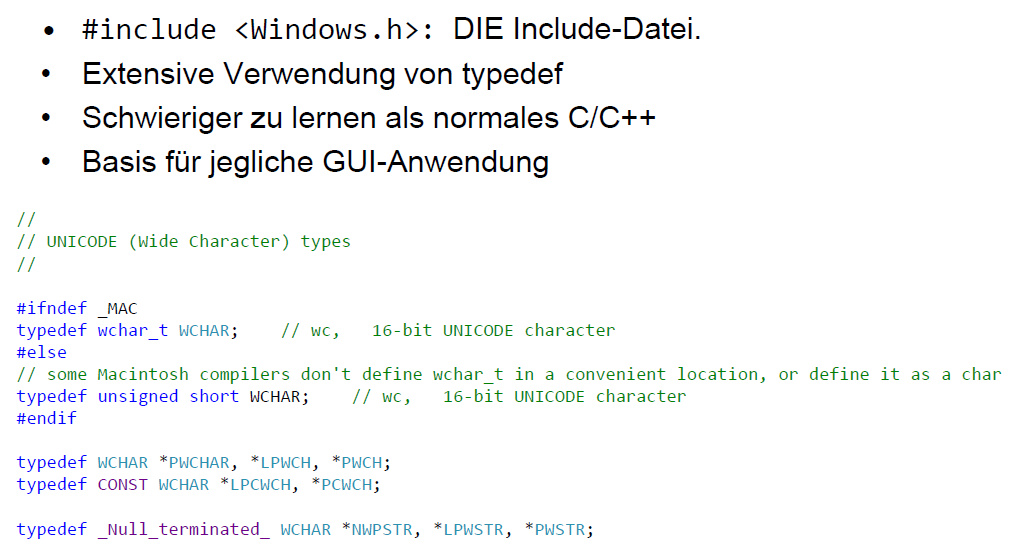




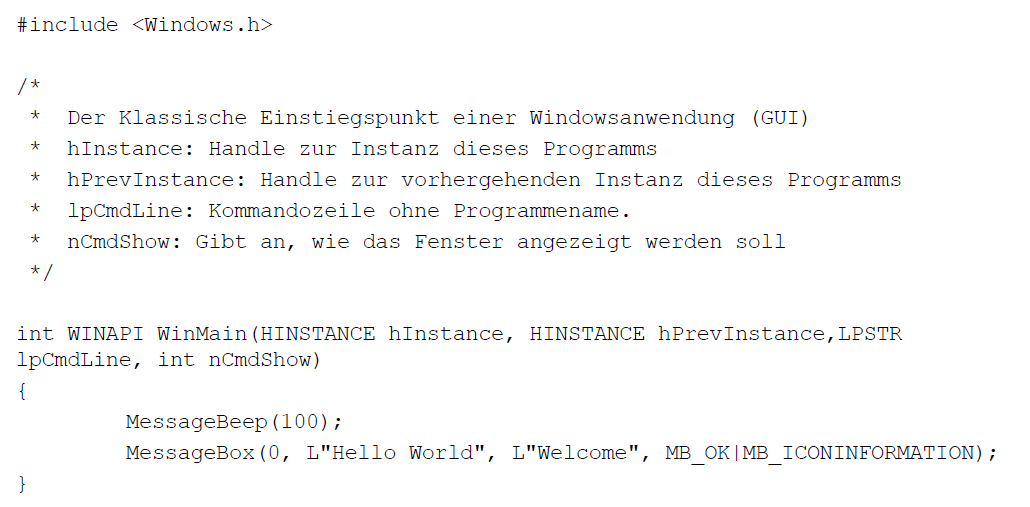
## Bemerkung zu Standard beim Programmieren



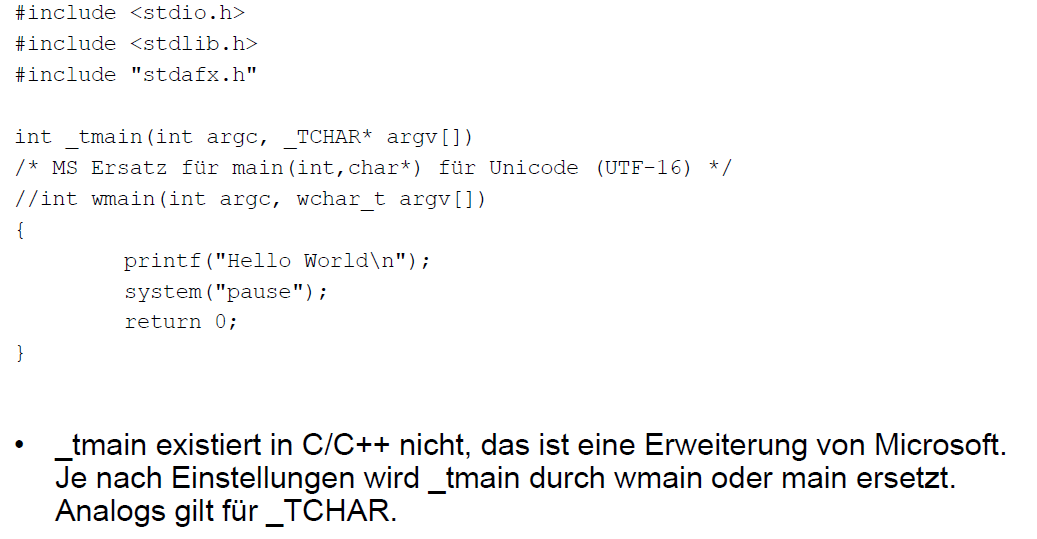
## Win32 API und C Syntax

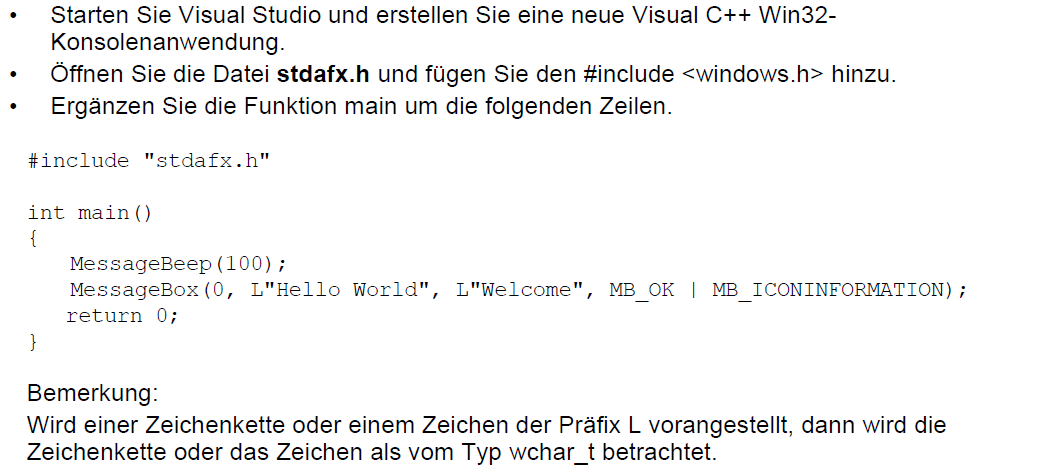


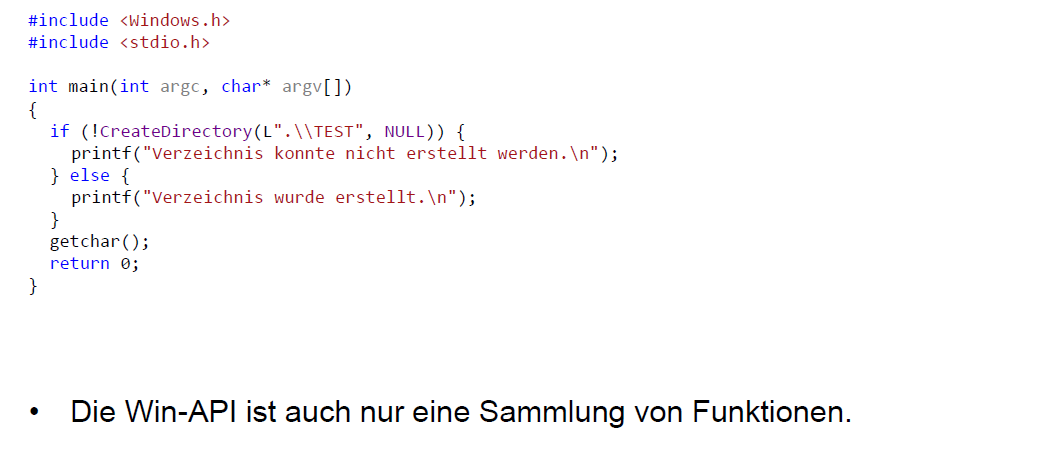
## WinMain als Einstiegspunkt für Windows-Programme



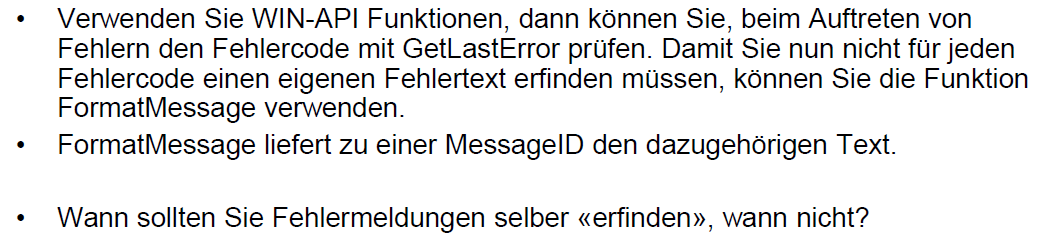
## Win32 Konsolenprogramm mit Visual Studio

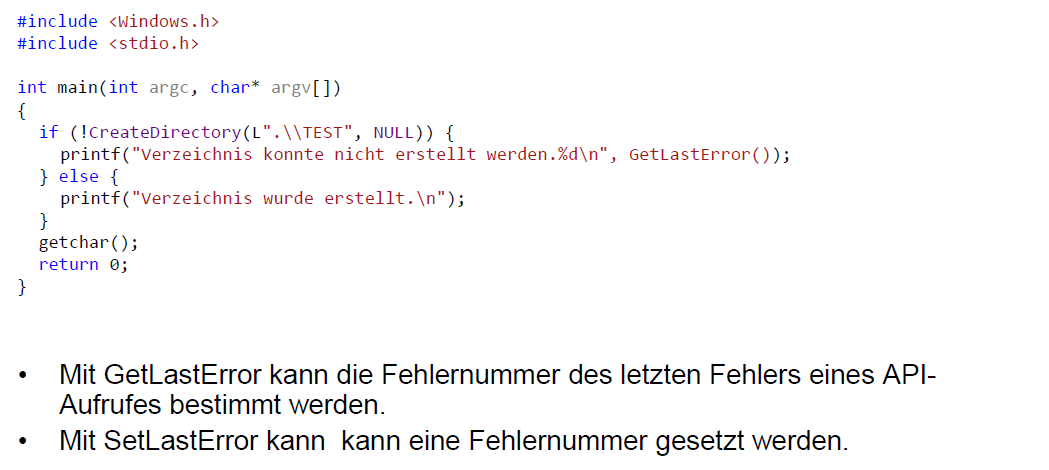


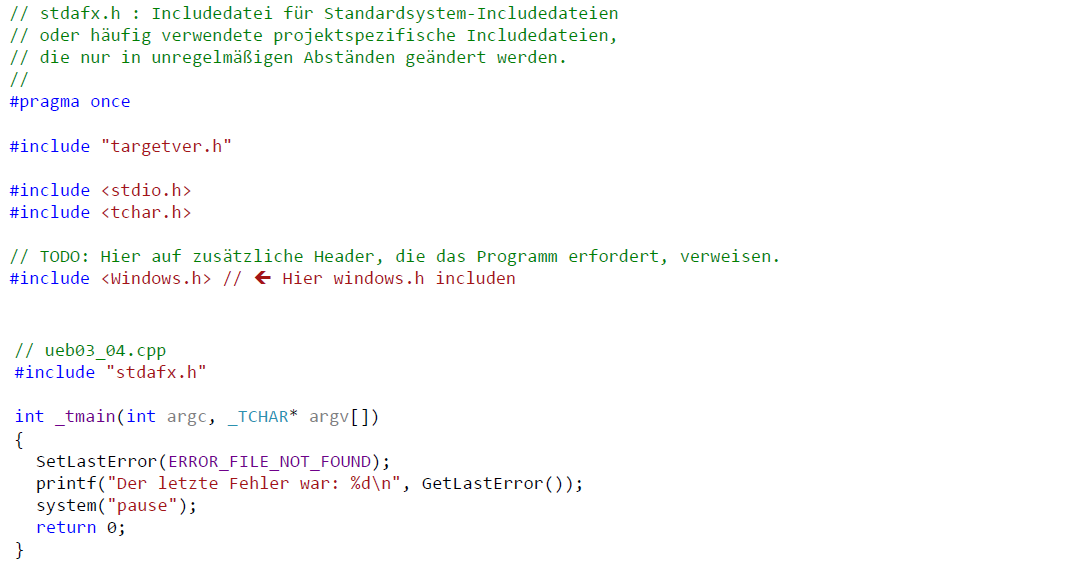




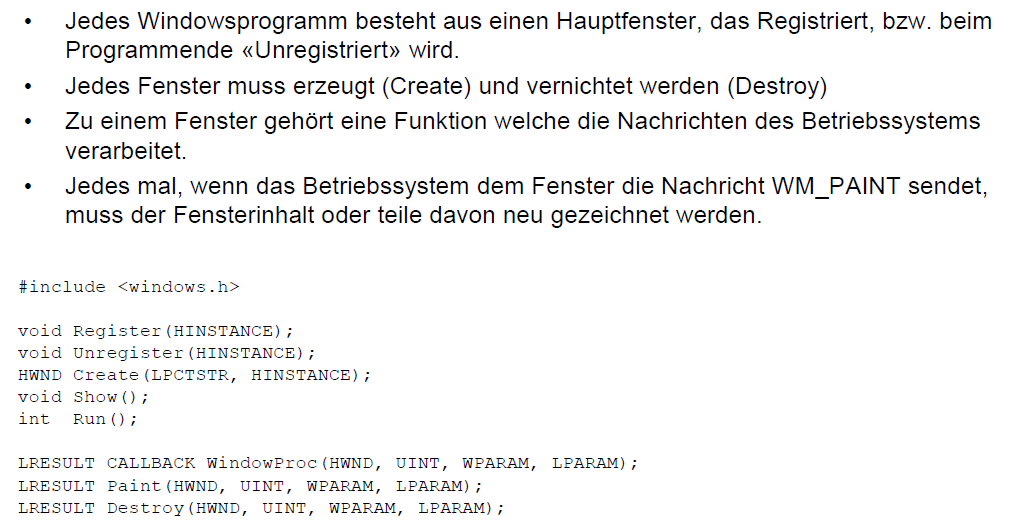
## Fehlermeldungen mit FormatMessage und GetLastError

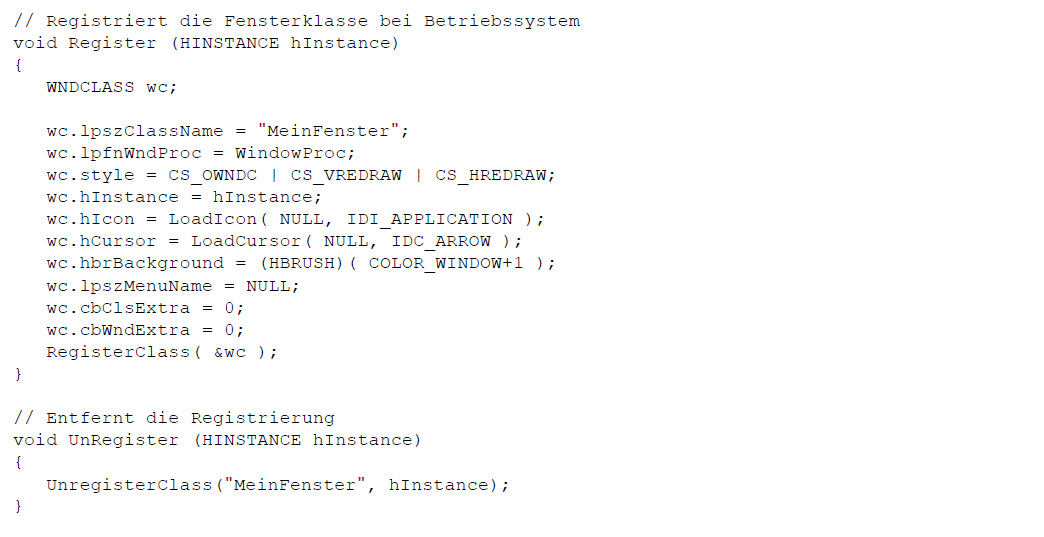






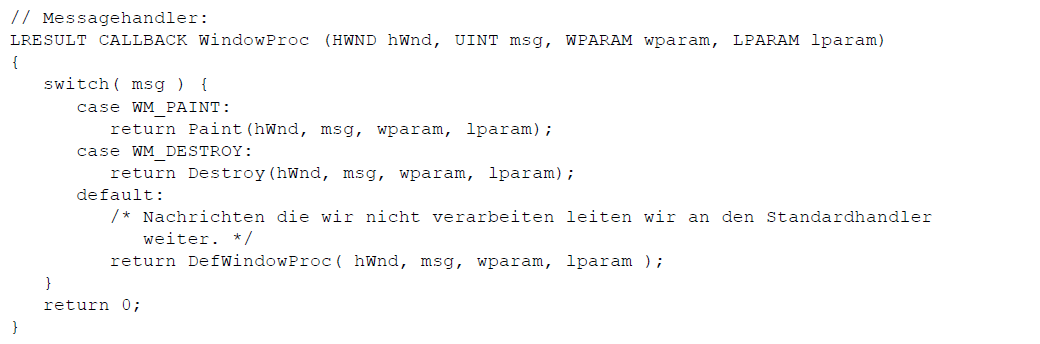
## Win32 Applikation mit Visual Studio

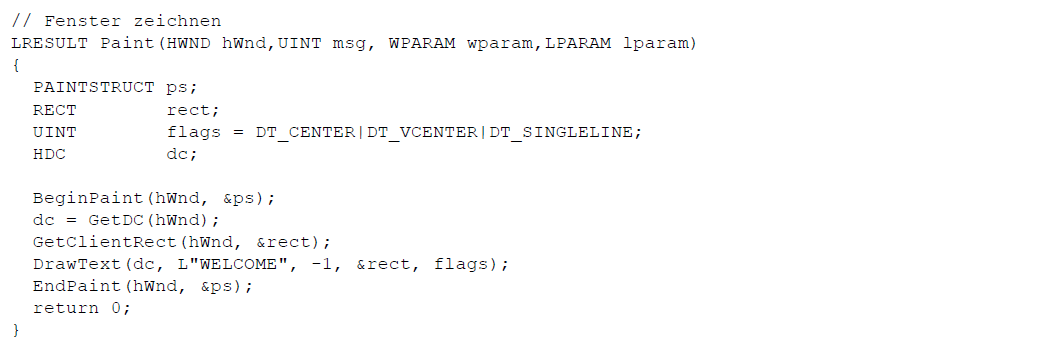


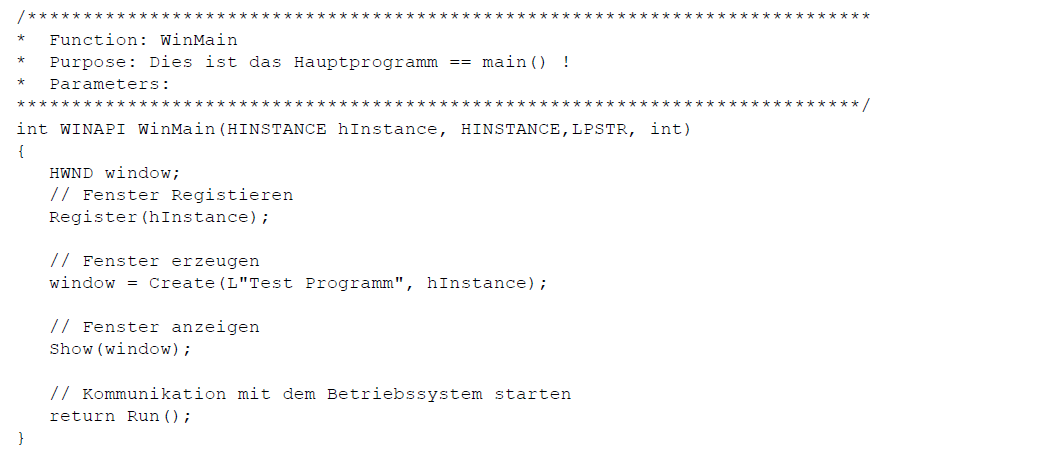




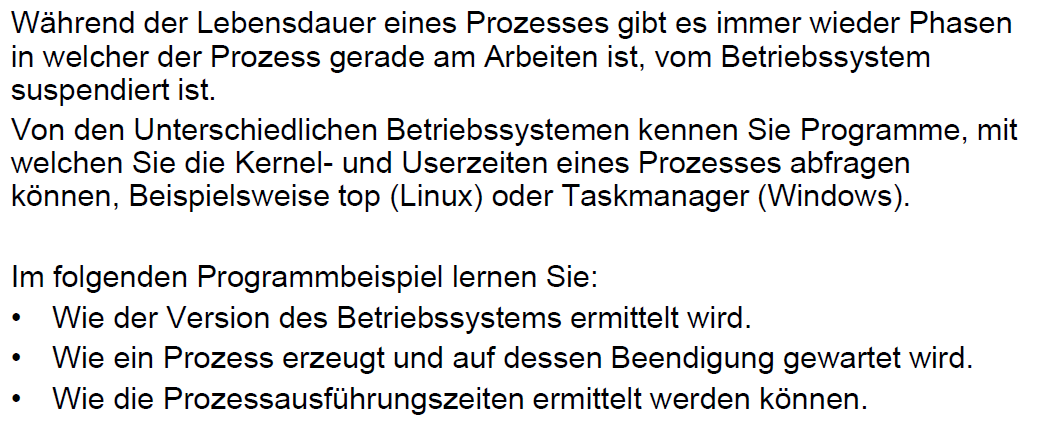


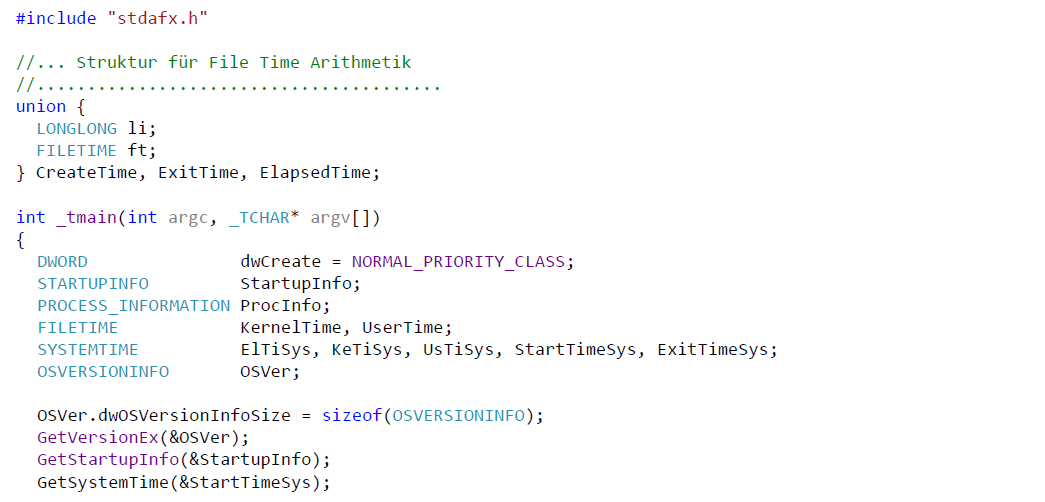


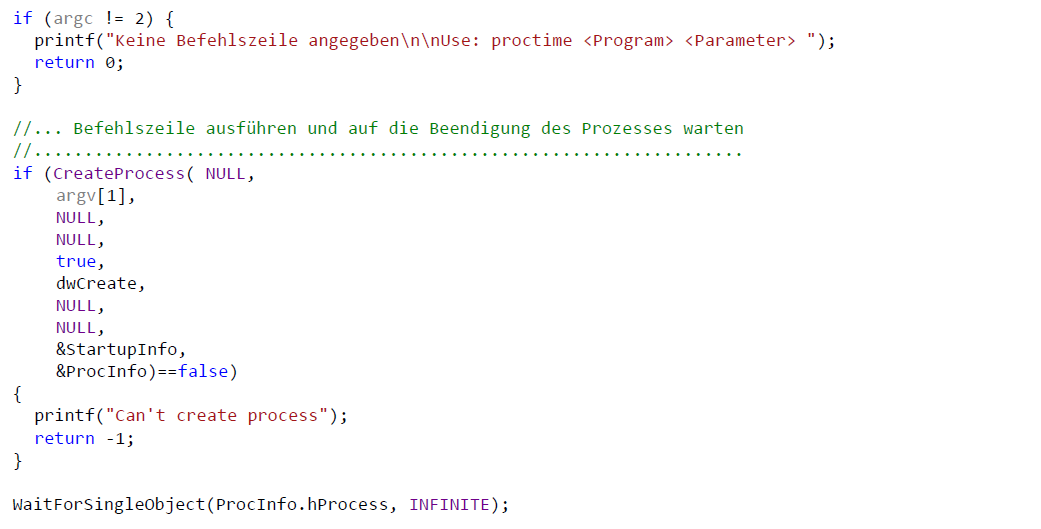


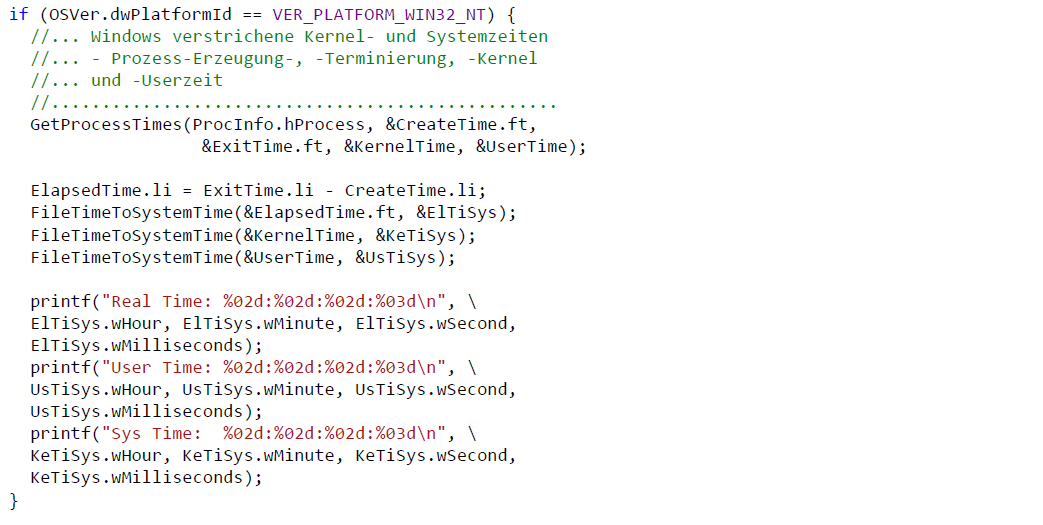


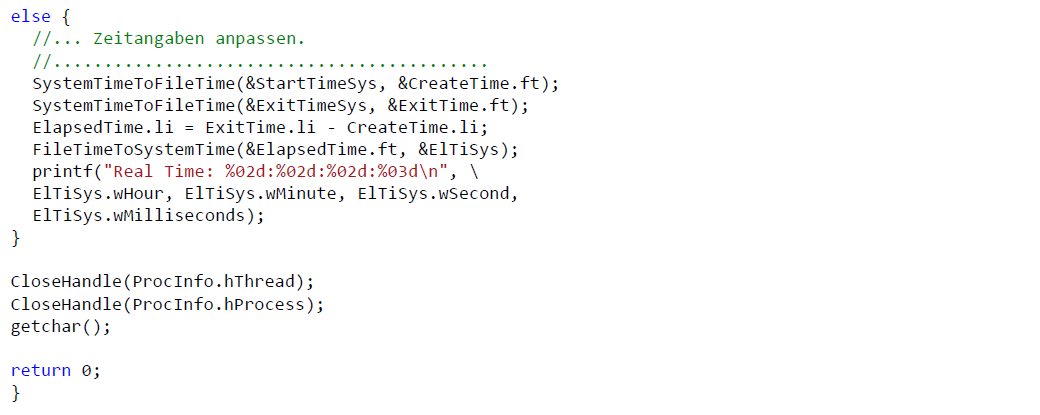
## Prozesszeiten





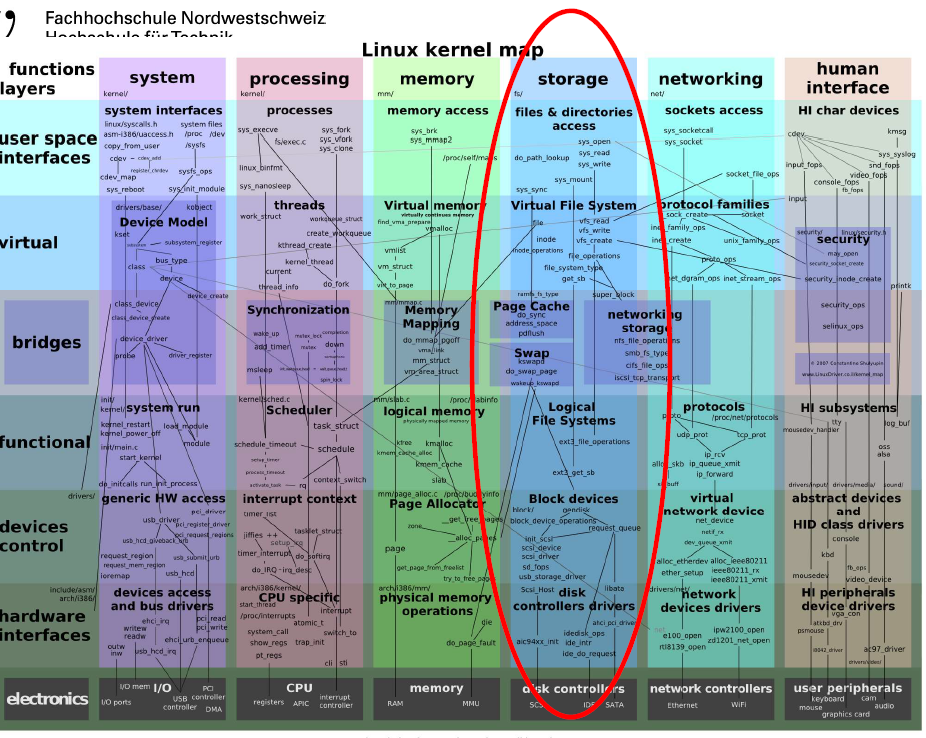




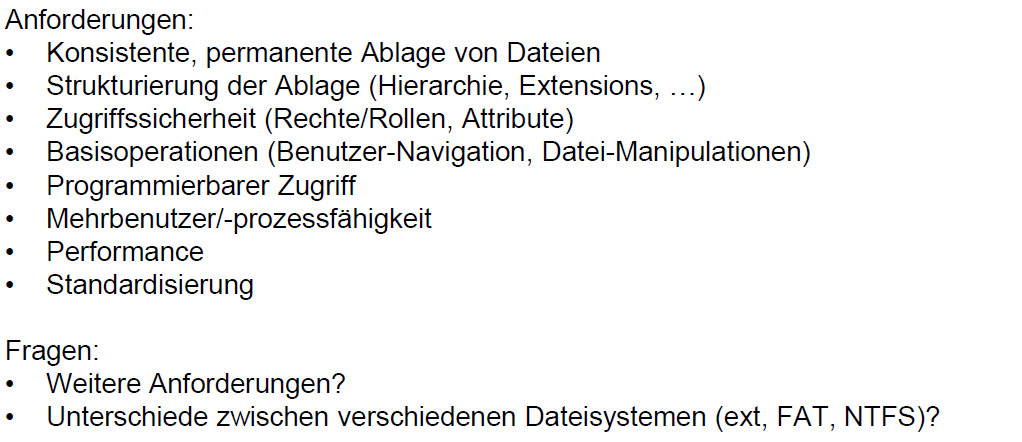


# Woche 5

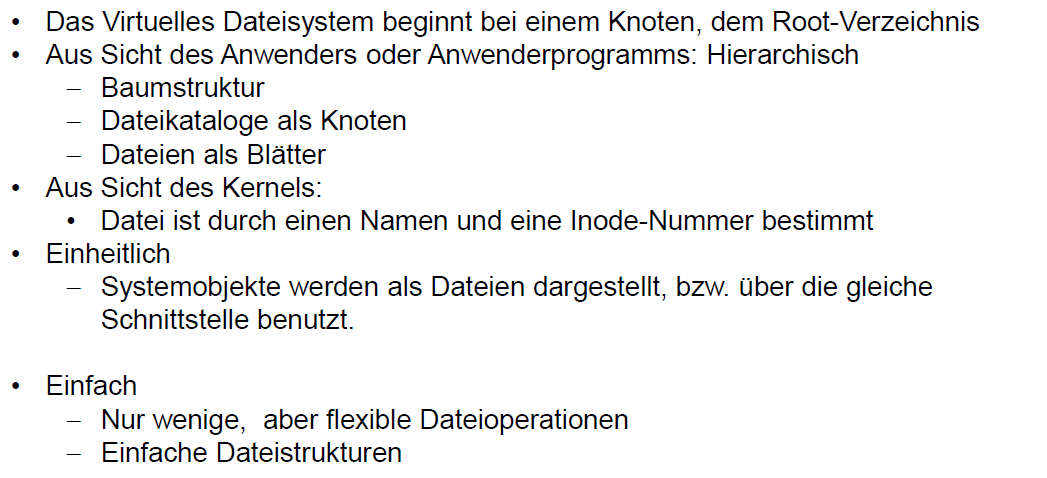
## Kontext von Dateisystemen



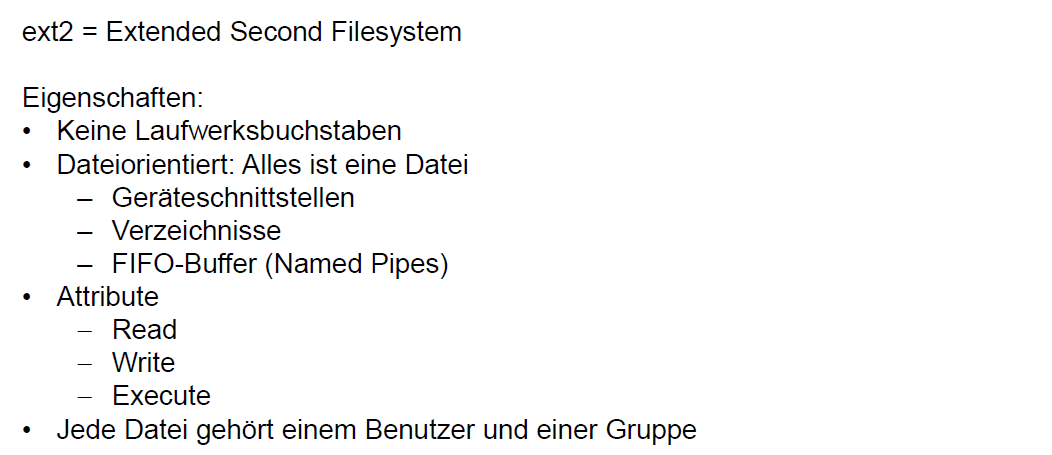
## Anforderungen an ein Dateisystem



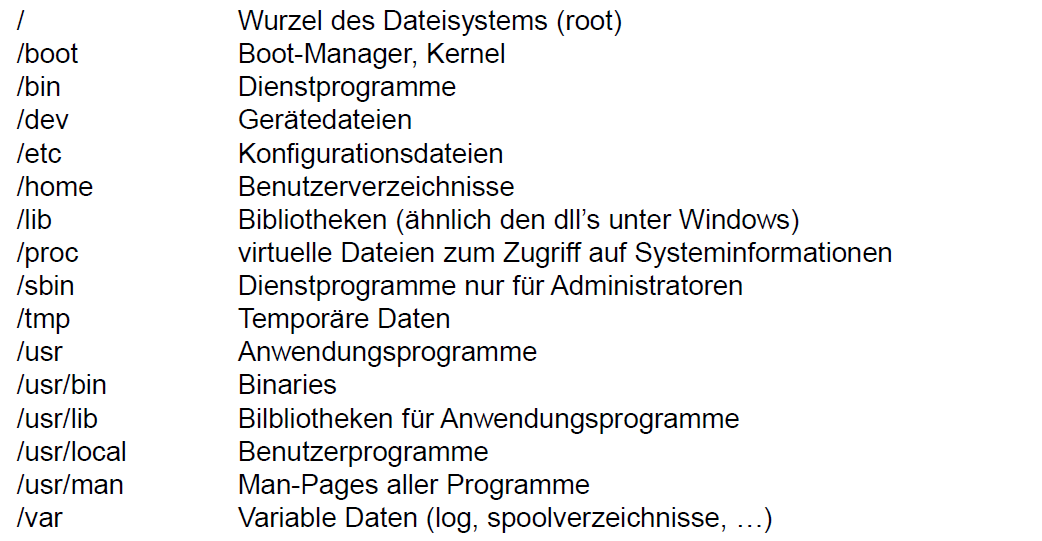
## Aufbau eines Unix Dateisystems



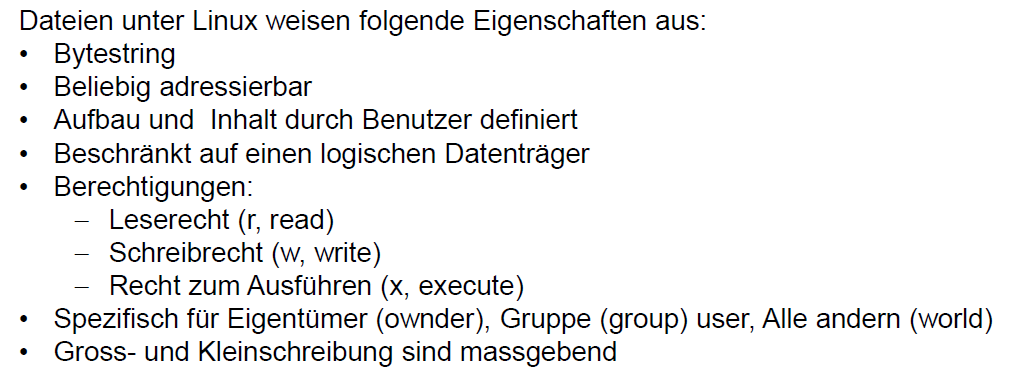
## Beispiel Linux Dateisystem ext2



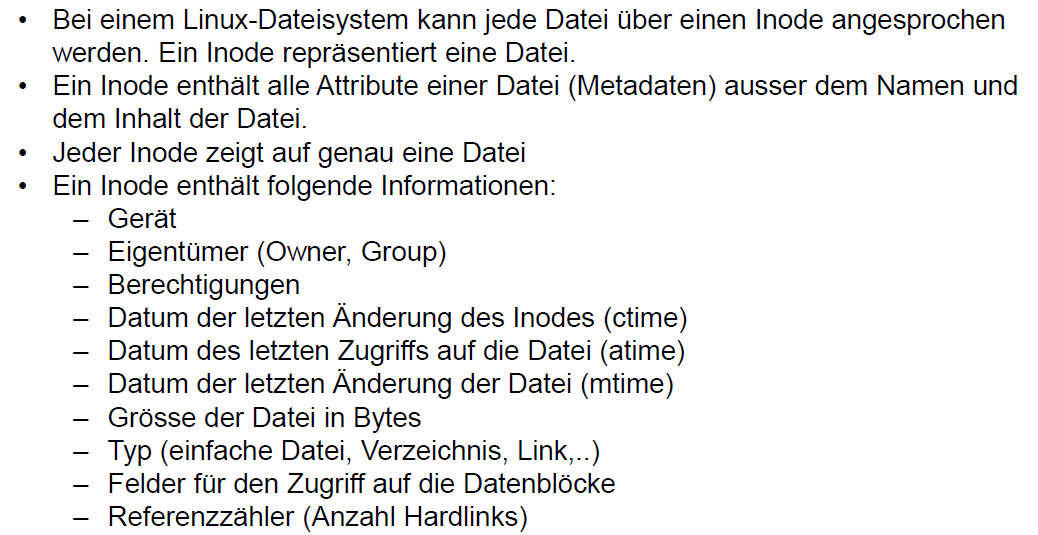
## Gliederung von ext2



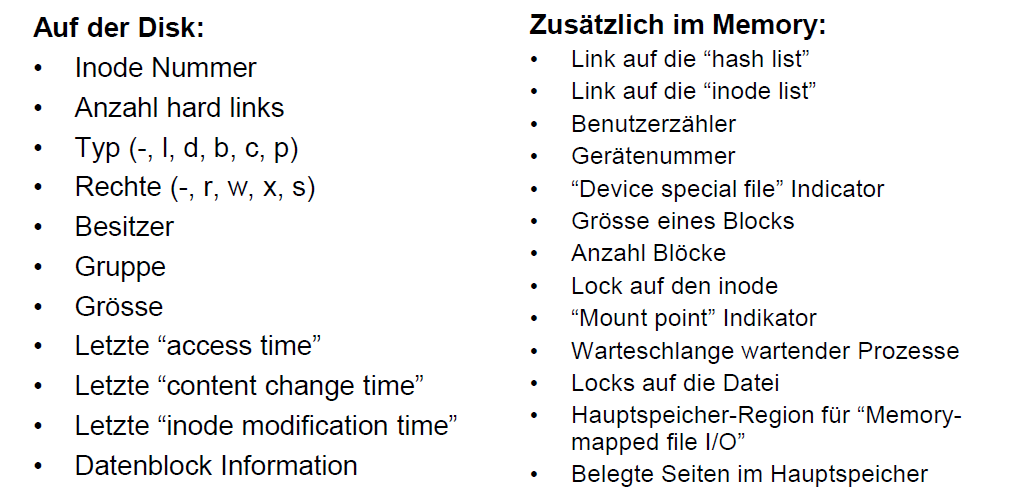
## Dateien unter Linux



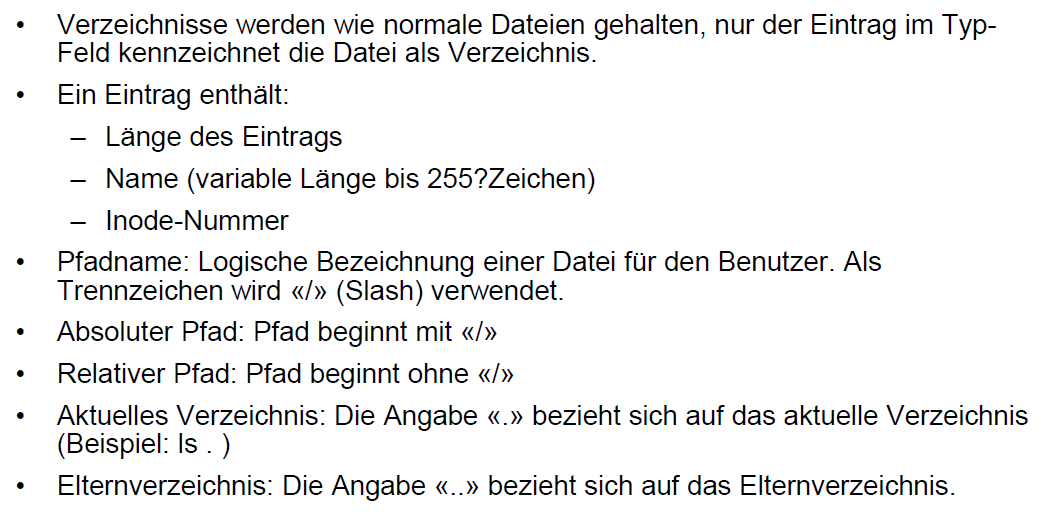
## Index Nodes als sogenannte Inodes



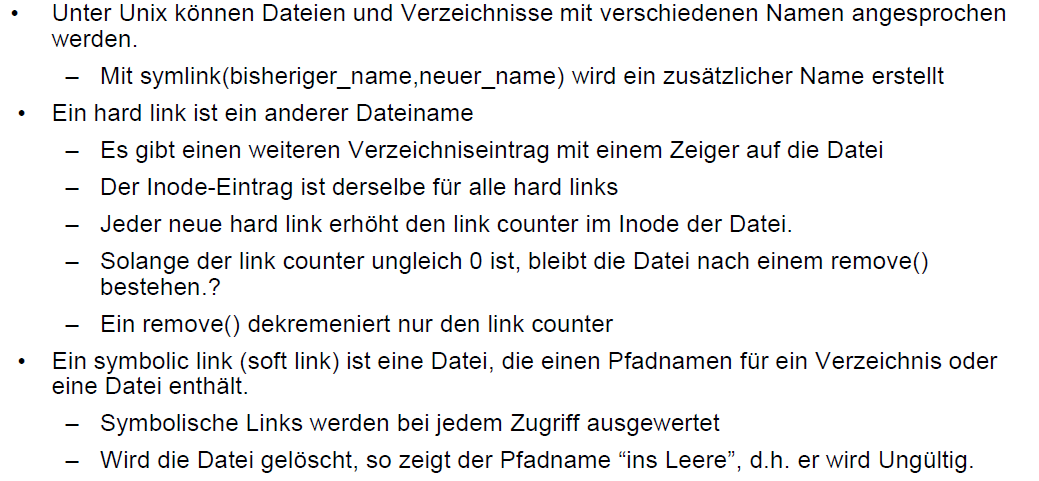
## Design und Aufbau einer Inode



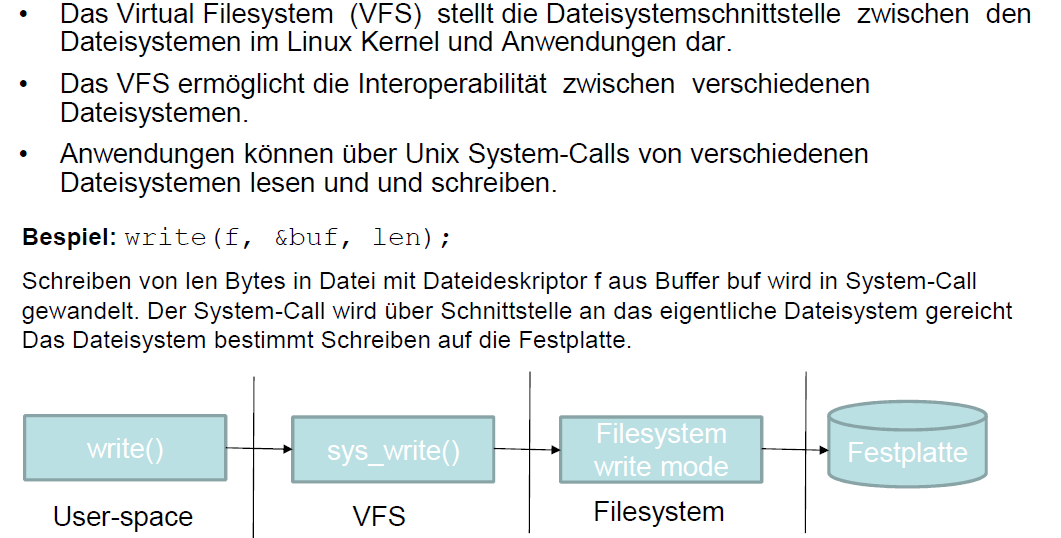
## Verzeichnisse



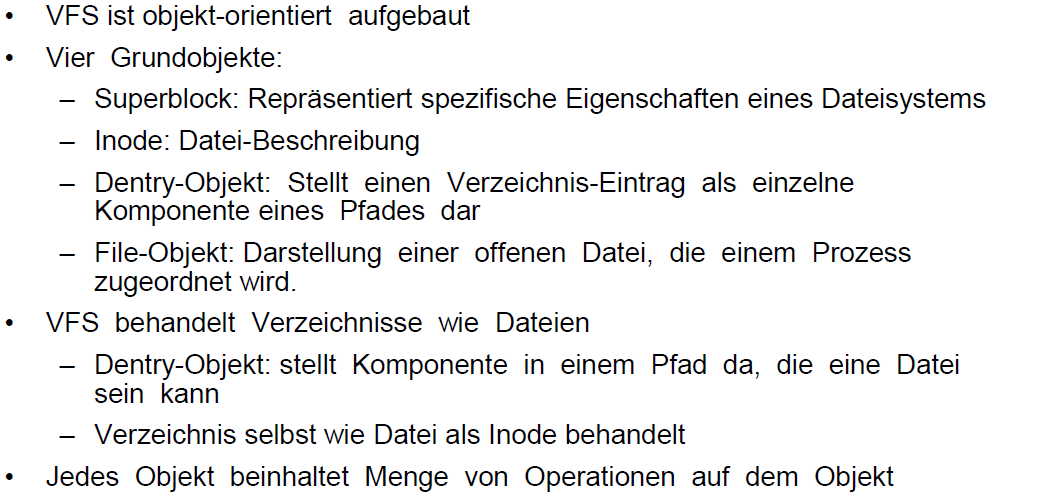
## Hard- und Symbolic Links



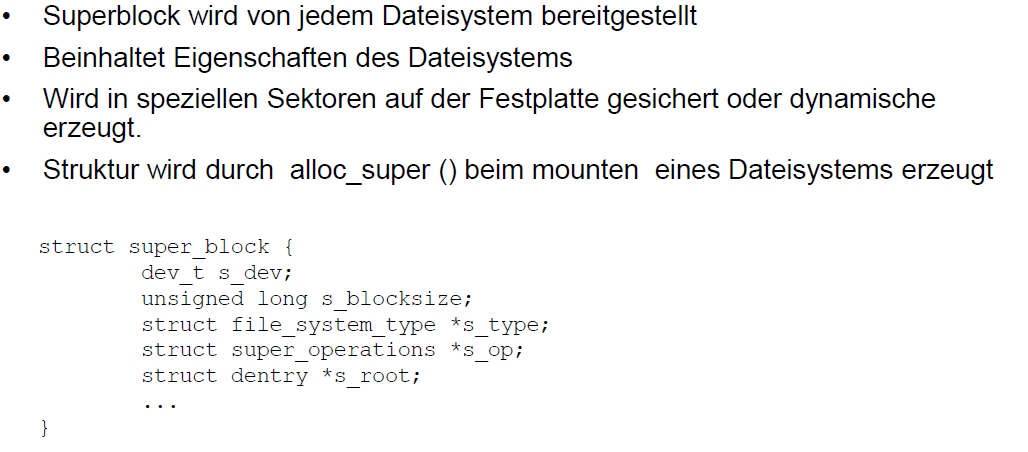
## Virtuelles Dateisystem



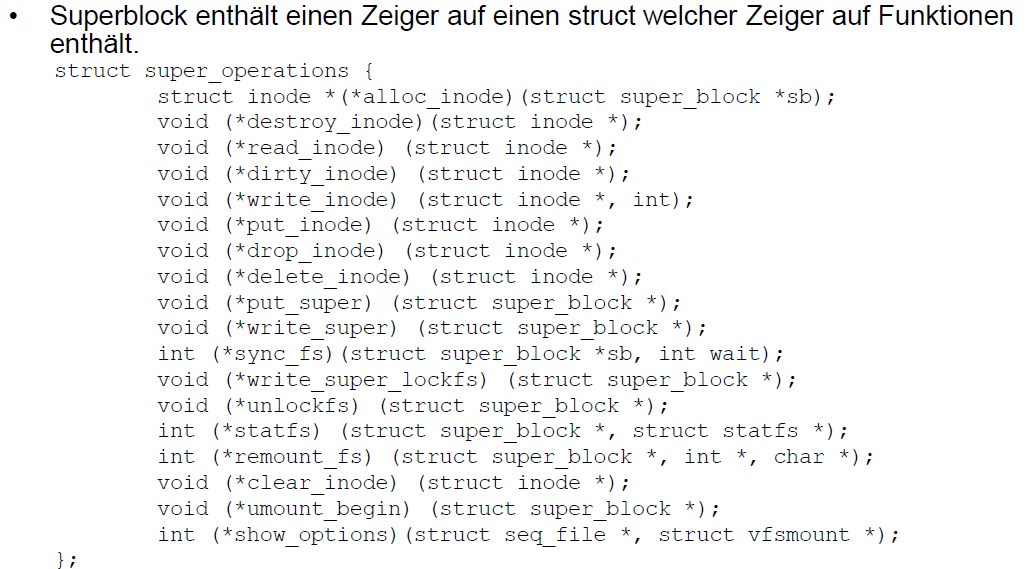
## VFS Objekte und Datenstrukturen



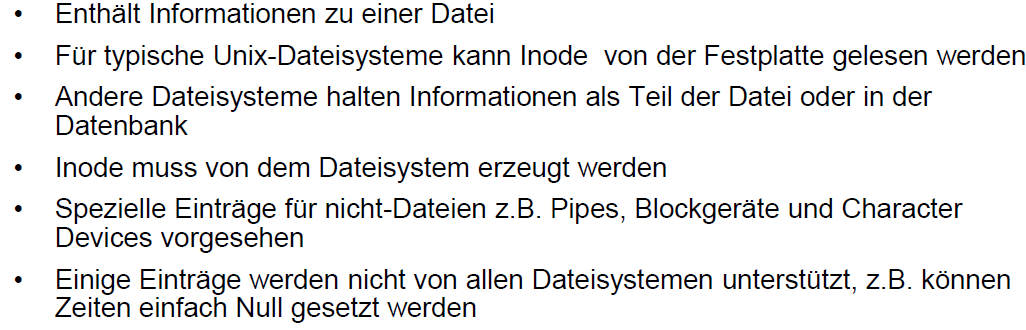
## Superblock



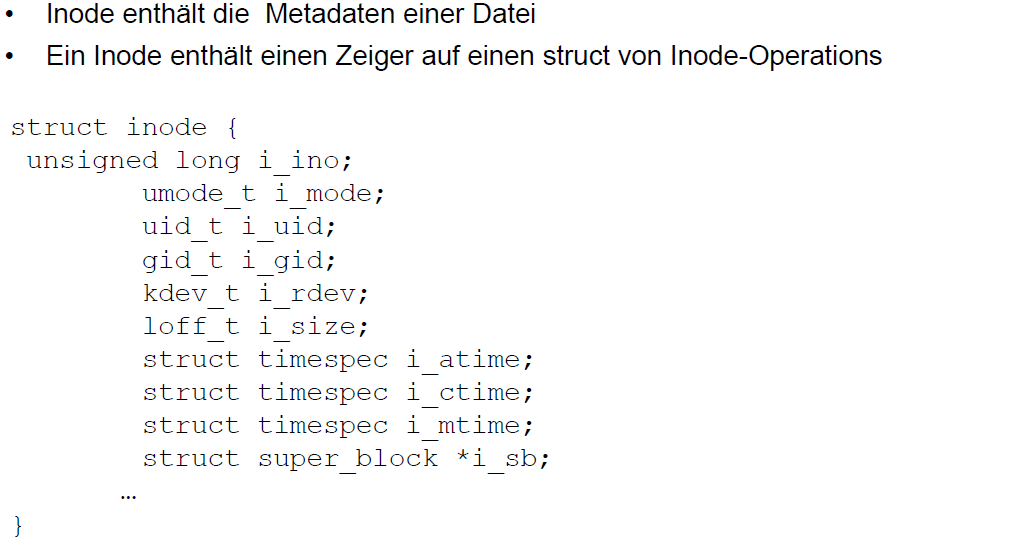
## Operationen auf Superblocks



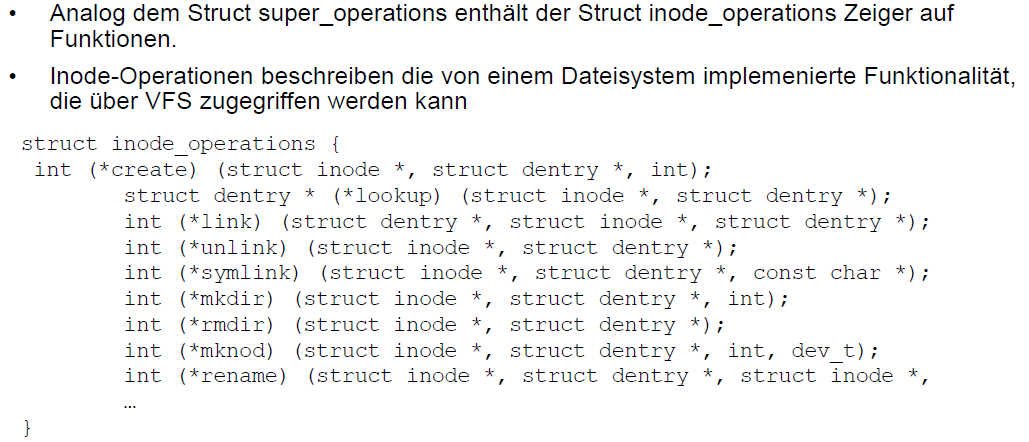
## Inodes



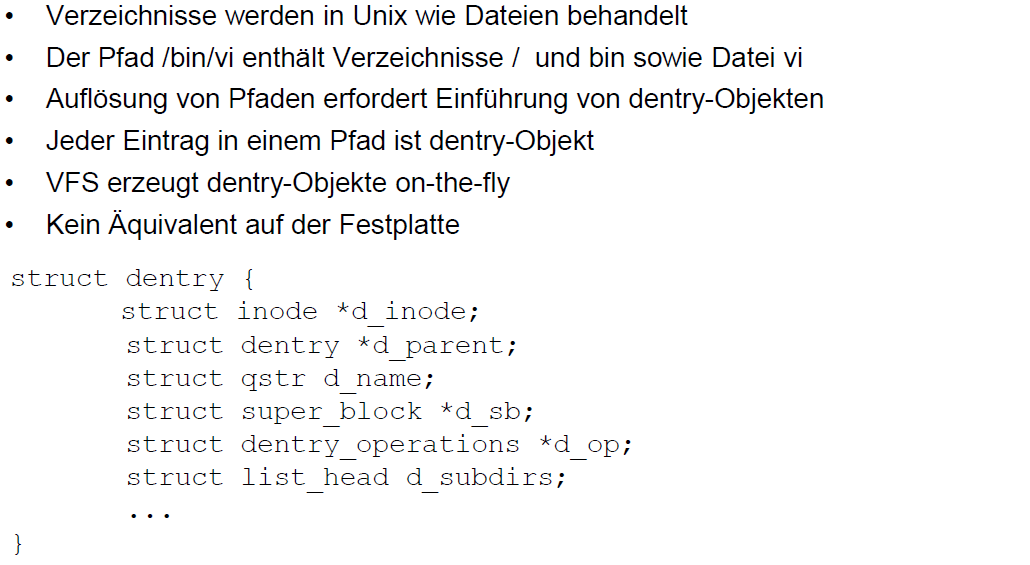
## Struktur struct inode



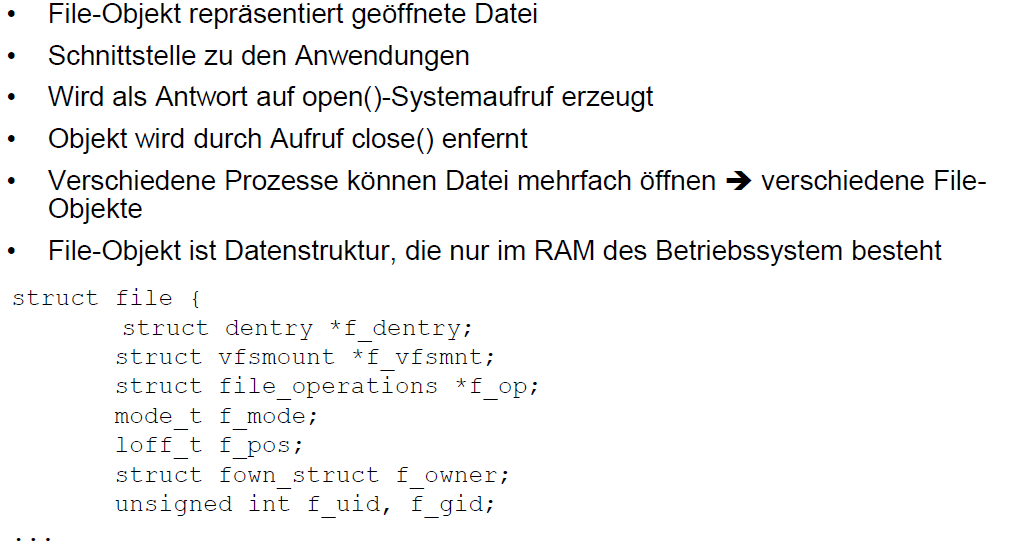
## Operationen auf inode



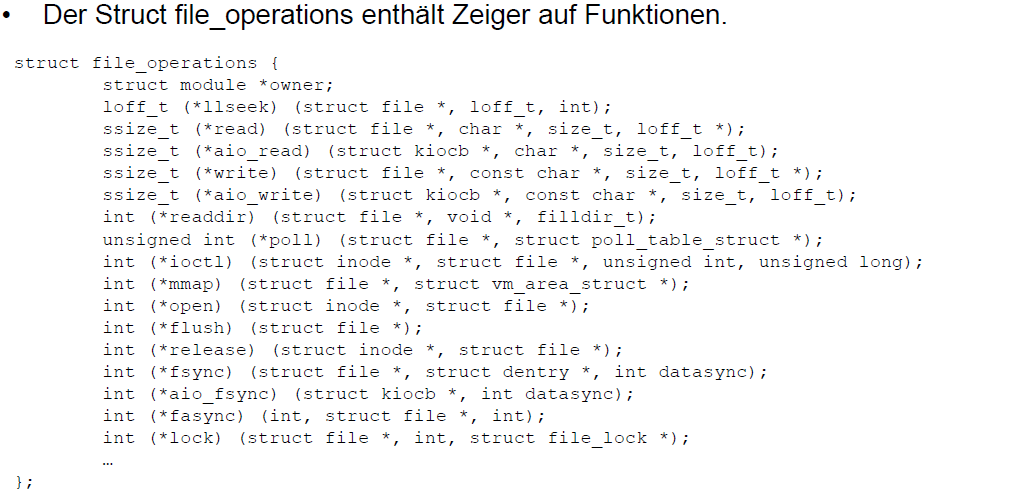
## Struktur struct dentry



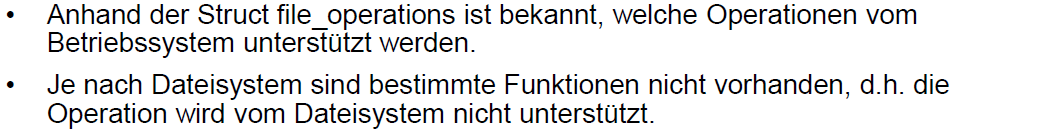
## Struktur struct file



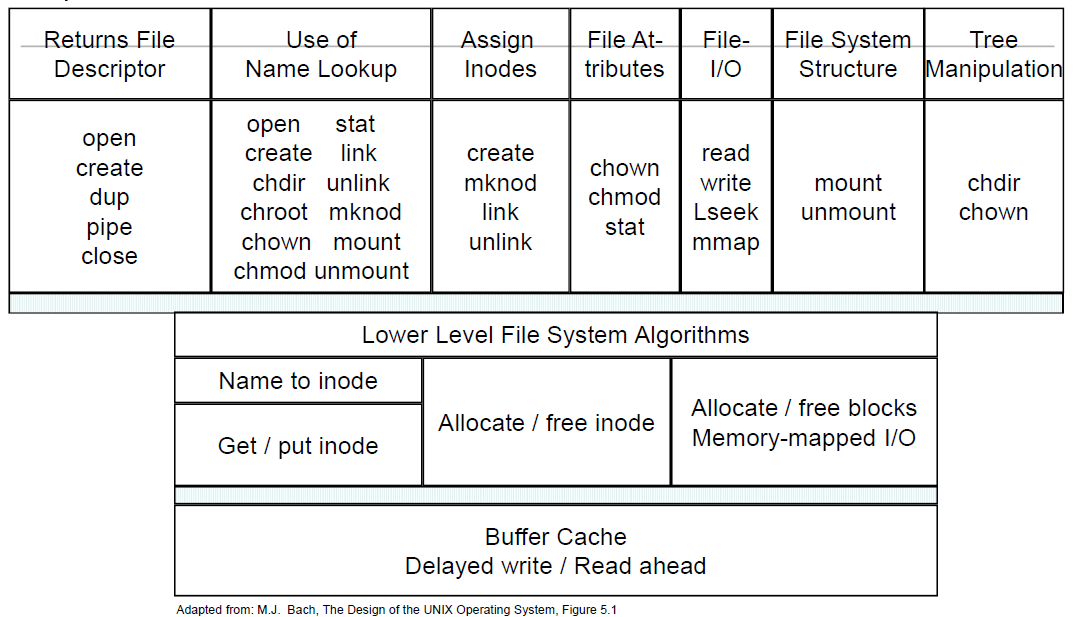
## Operationen auf file



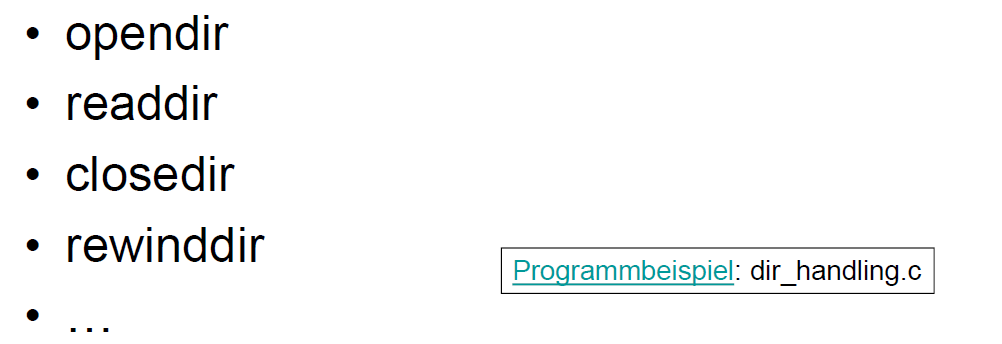
## Grundlegende Dateioperationen



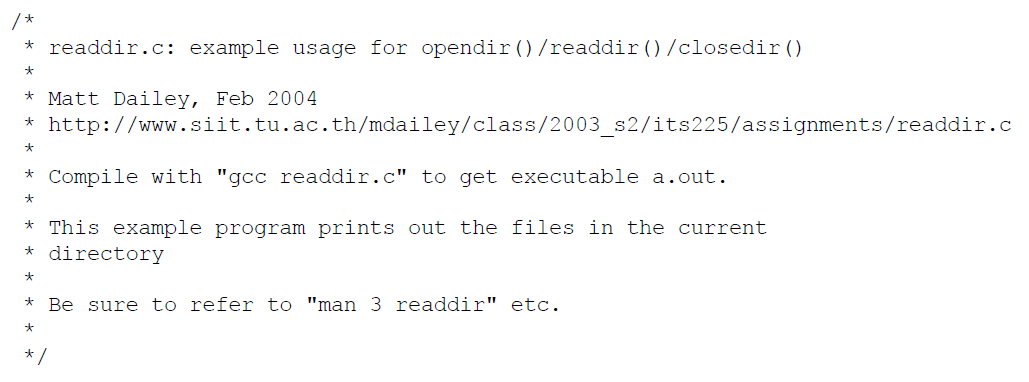
## System Calls im Linux Dateisystem

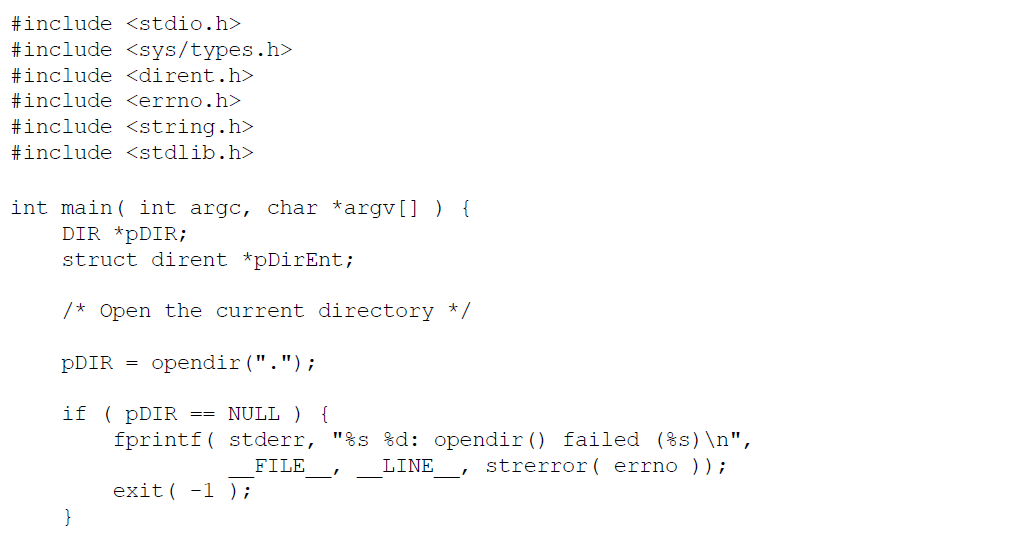


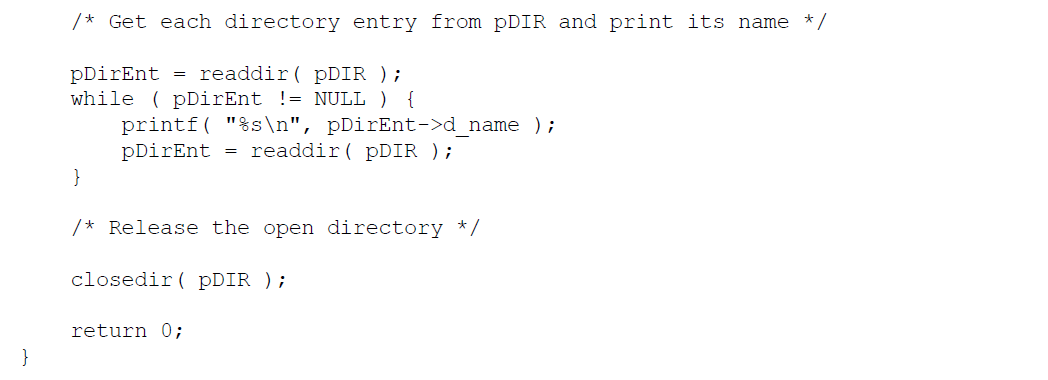
## Library Calls zur Verzeichnismanipulation



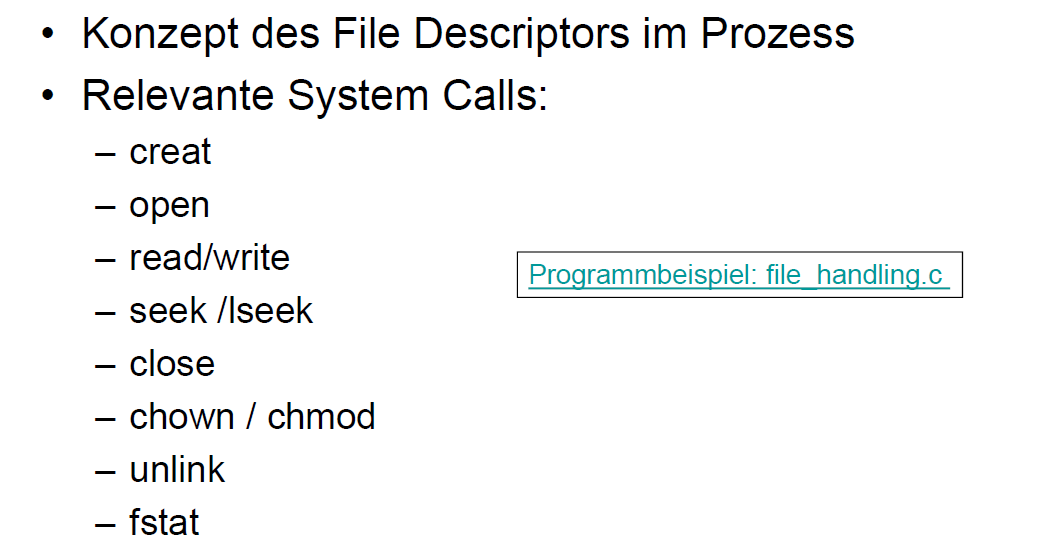
## Beispiel einer Verzeichnismanipulation



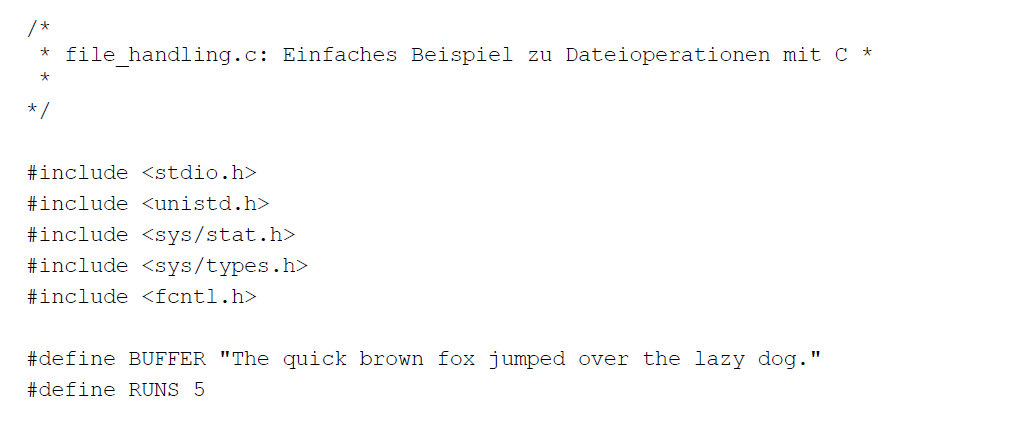


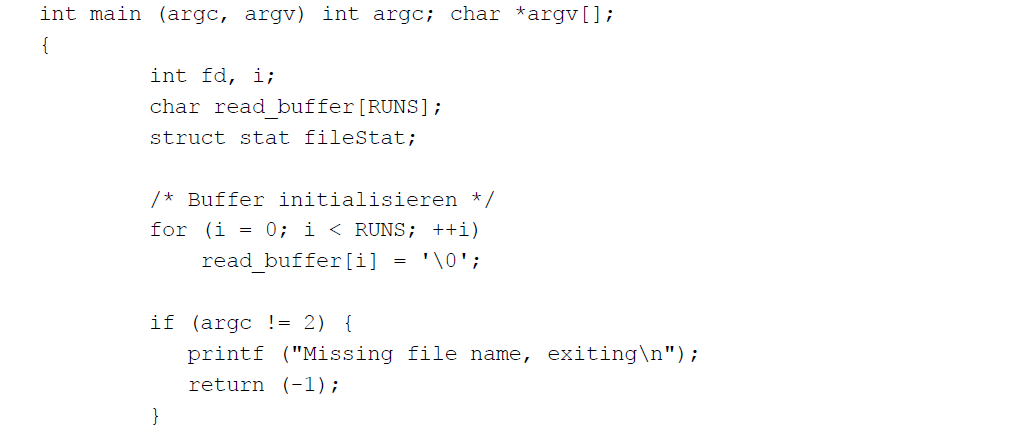


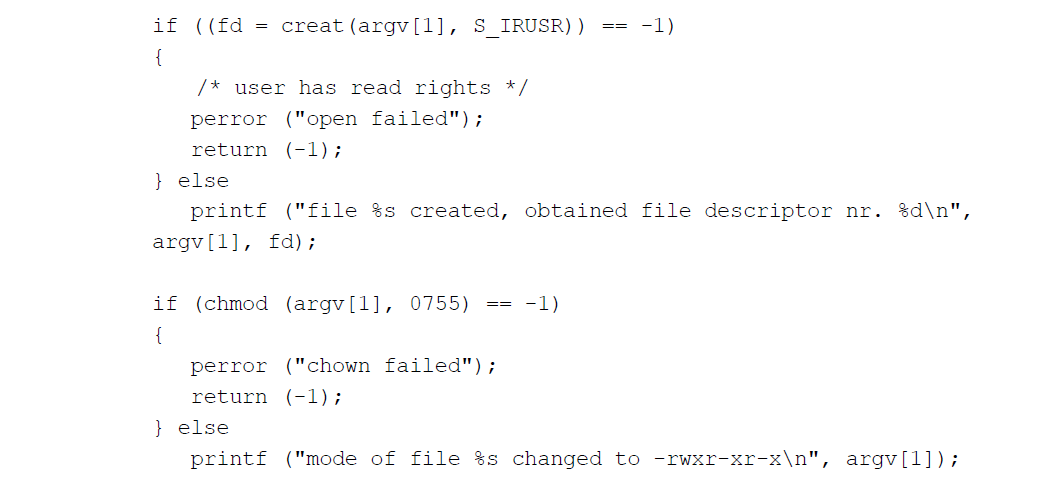
## System Calls zur Dateimanipulation

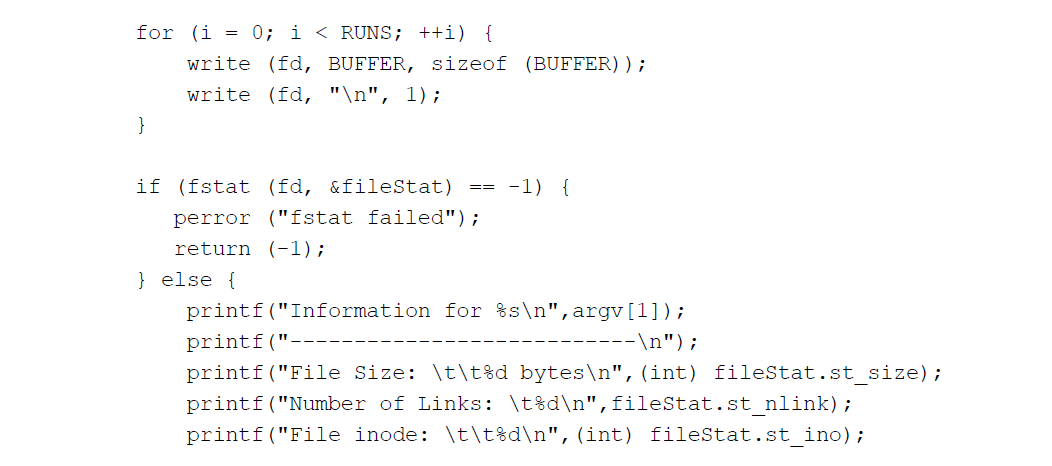


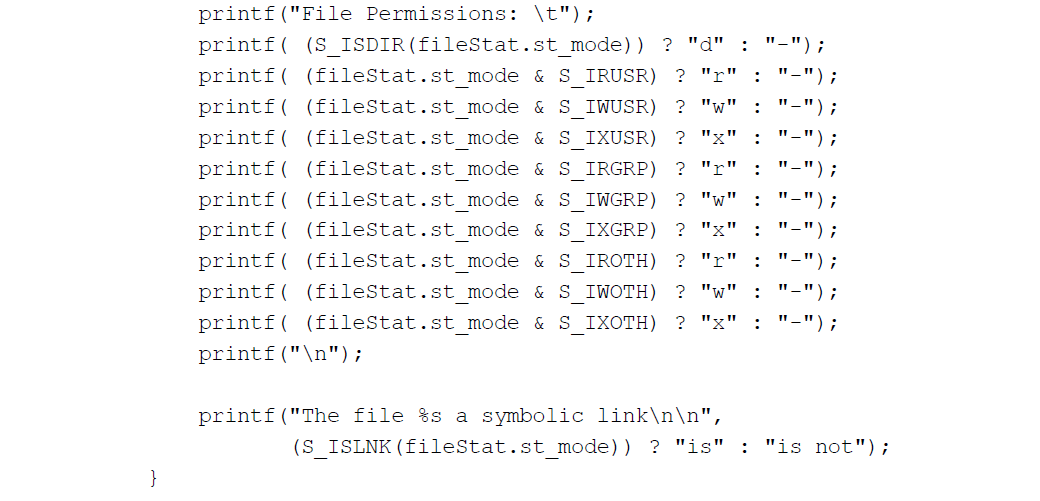
## Beispiel einer Dateimanipulation

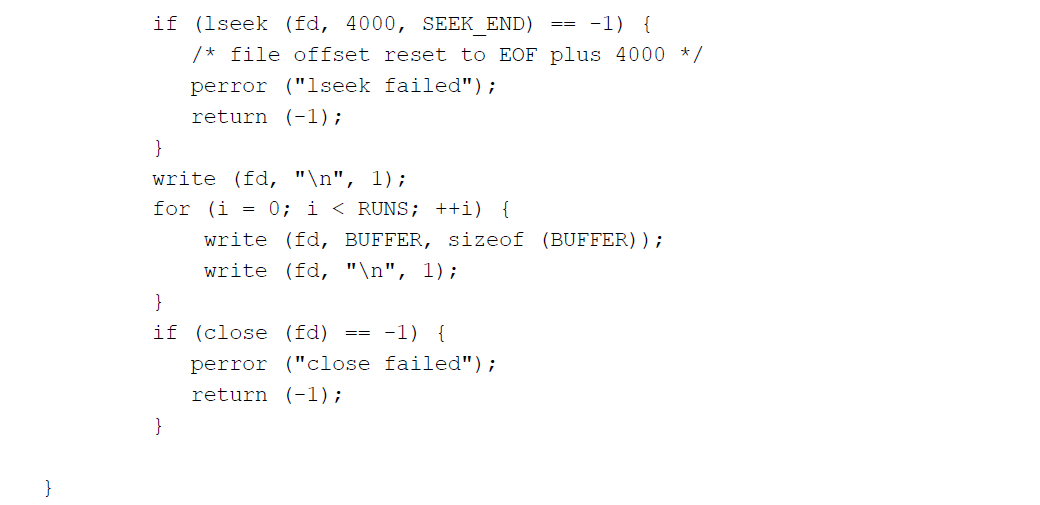








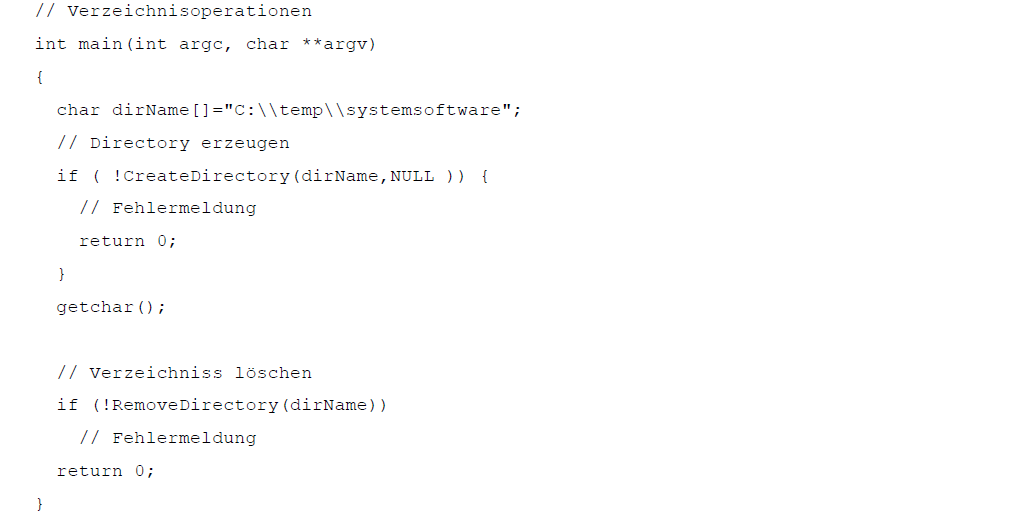


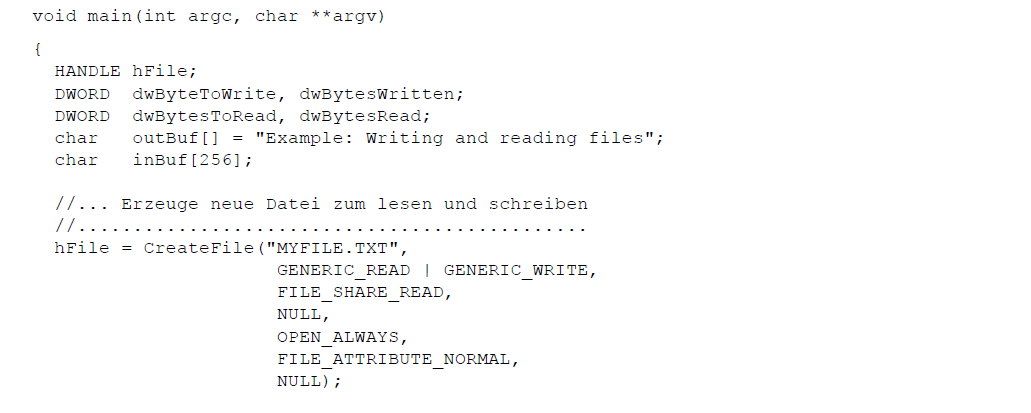


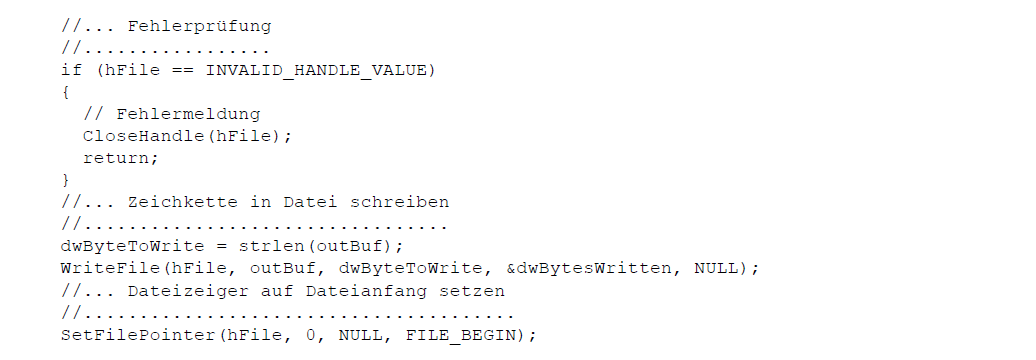
## System Calls zur Arbeit im Dateisystembaum

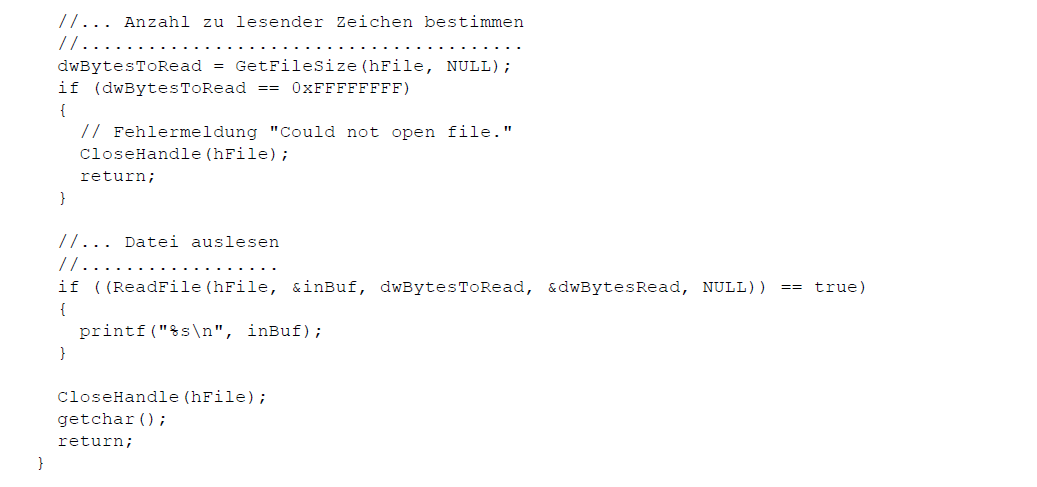


## Grundlegende API Funktionen in Windows



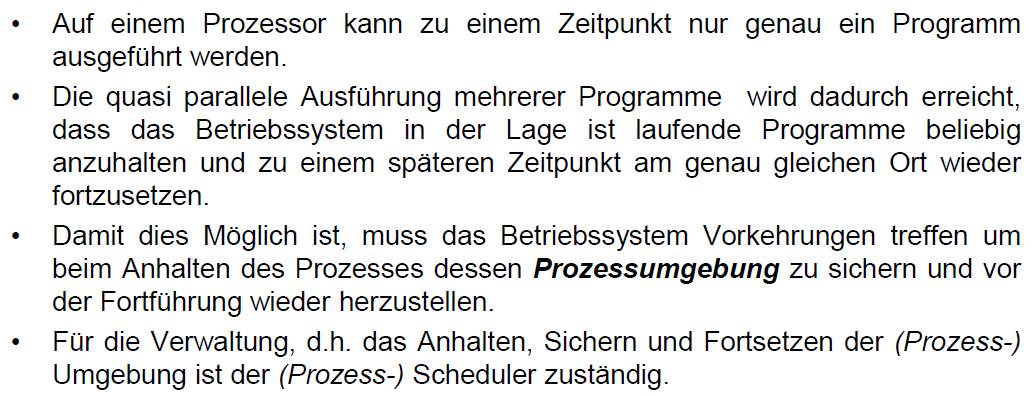




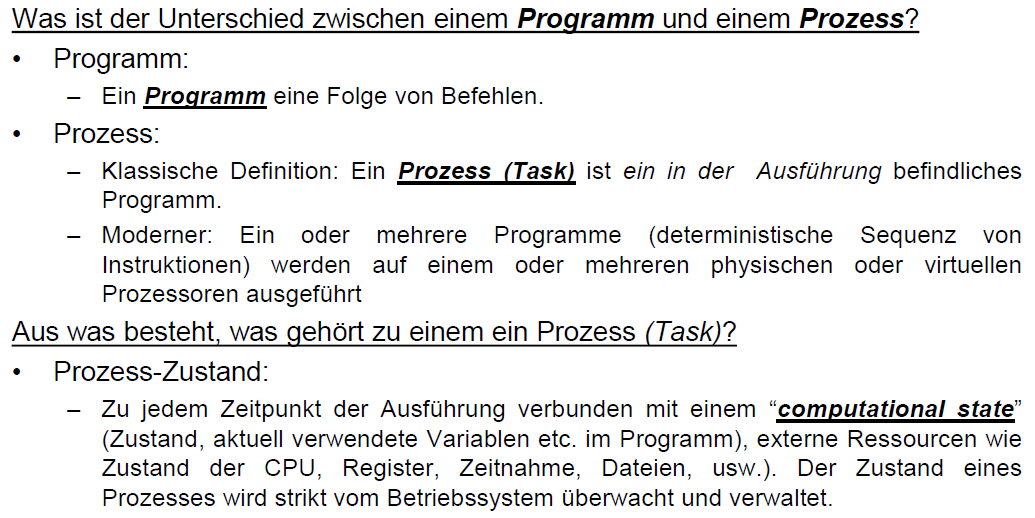


# Woche 6

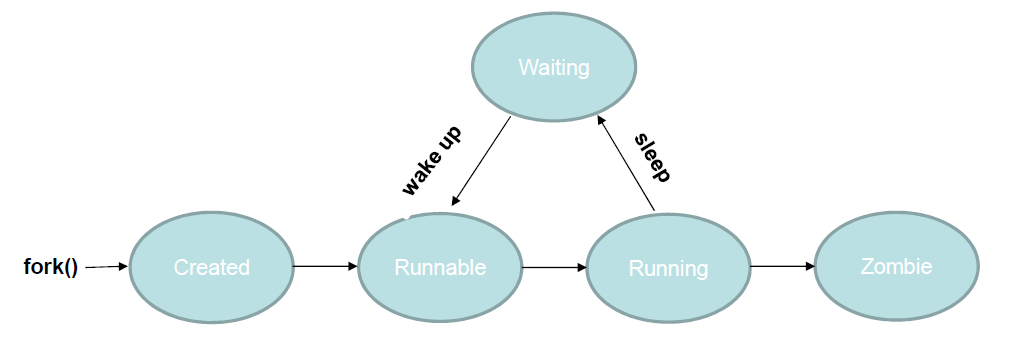
## Einführung in Prozesse

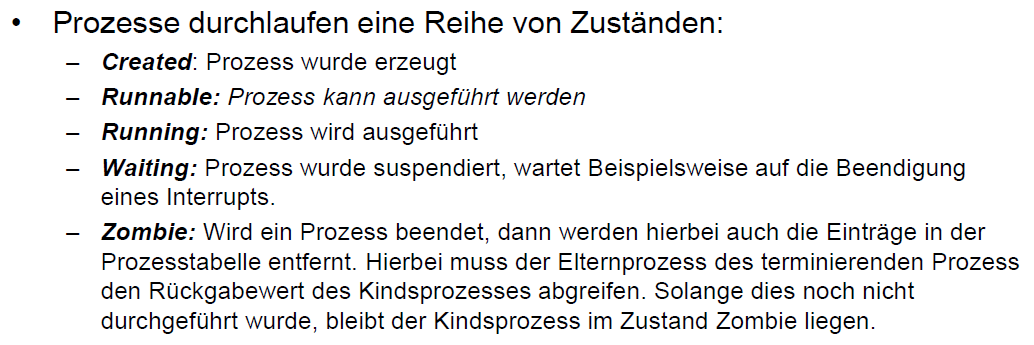


## Unterschied zwischen Prozesstask und Programm

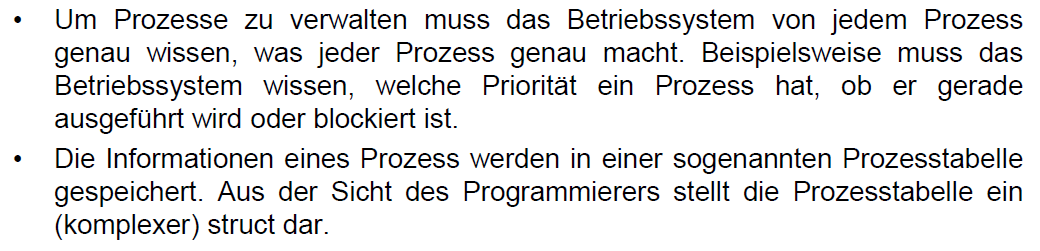


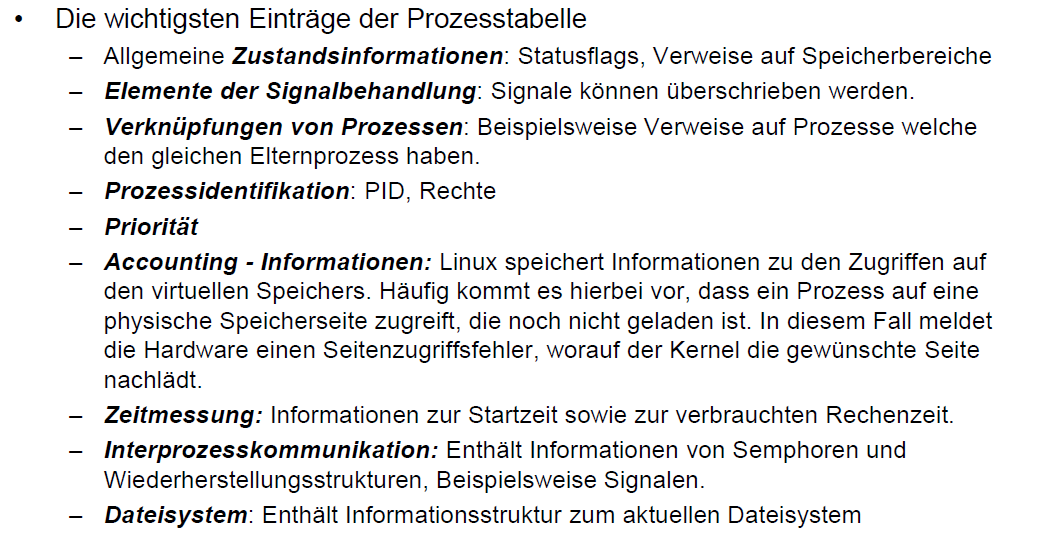
## Lebenszyklus eines Prozesses



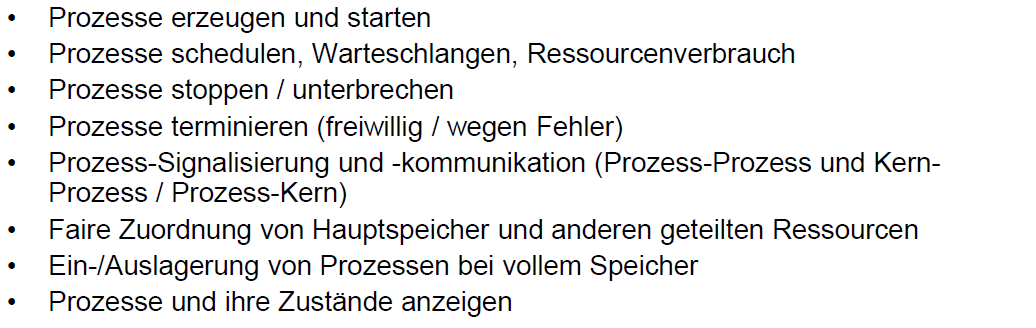


## Verwaltung von Prozessen

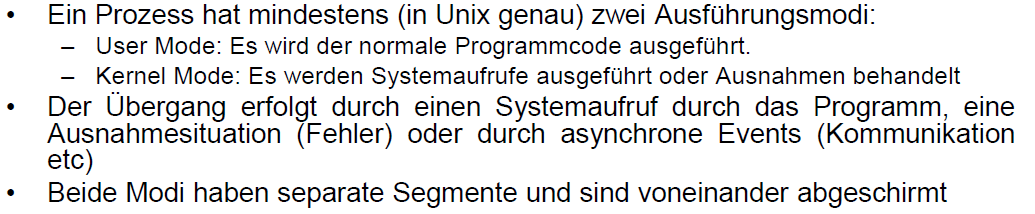




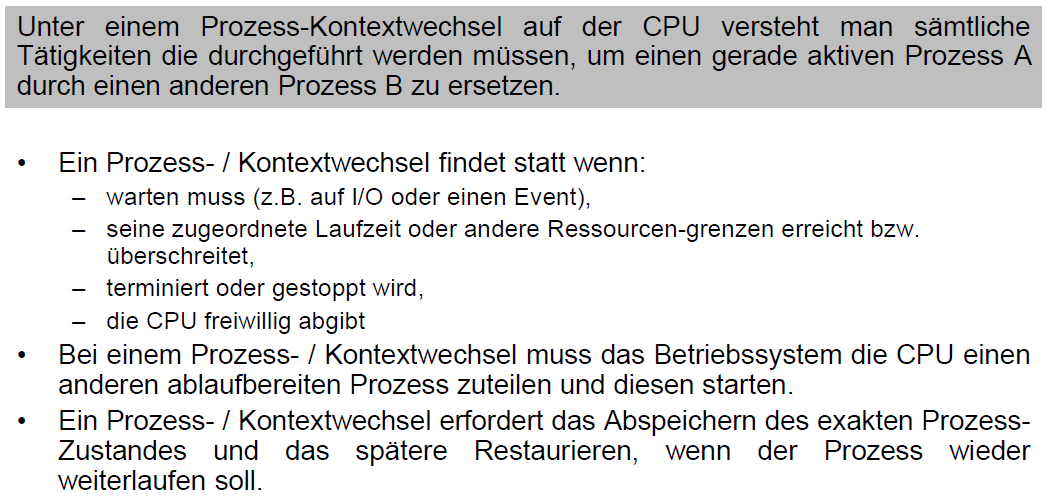
## Anforderungen an ein Prozesssteuersystem



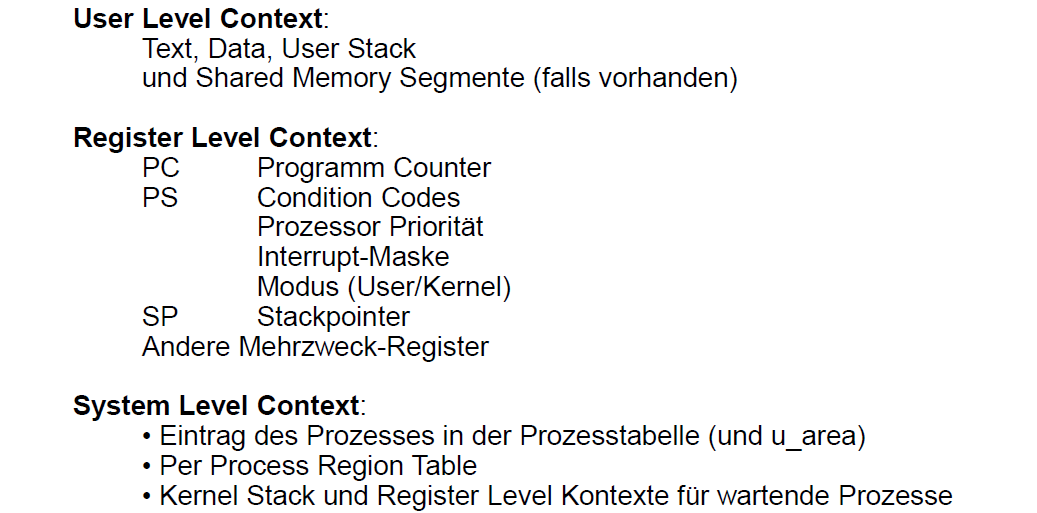
## User- und Kernel Mode



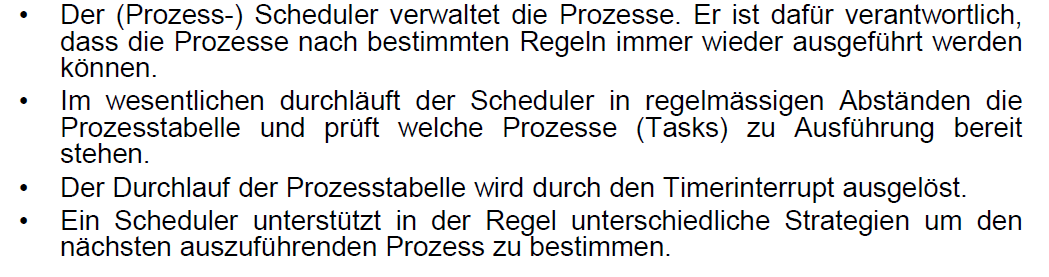
## Prozess-/Kontextwechsel



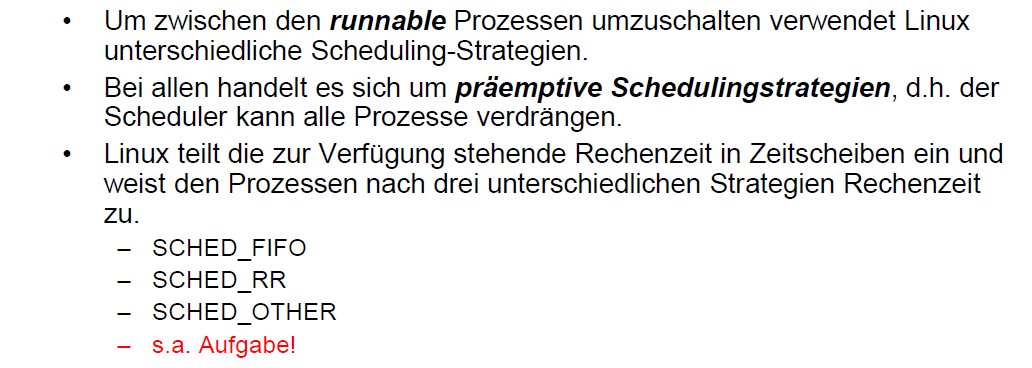
## Prozesskontext



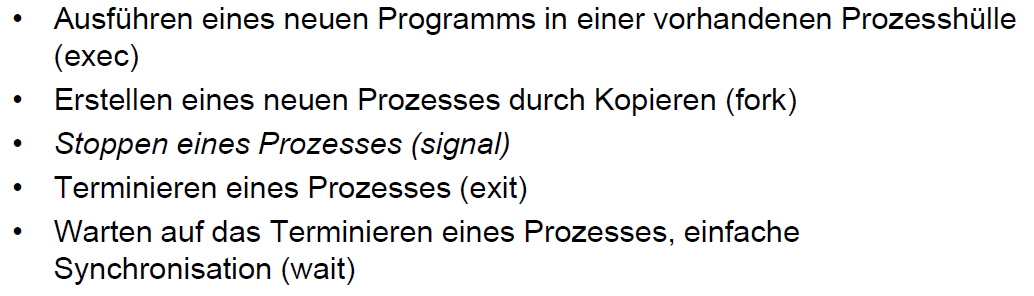
## Scheduler



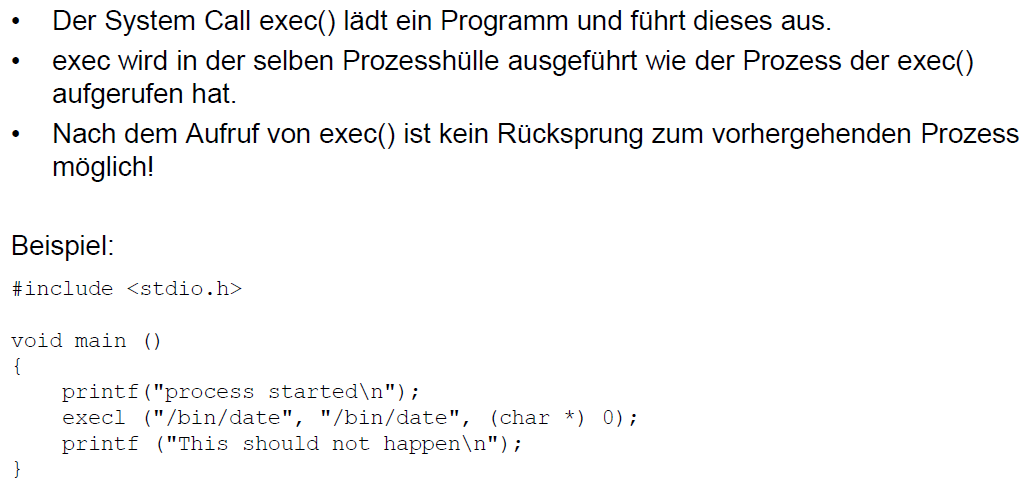
## Schedulingstrategien



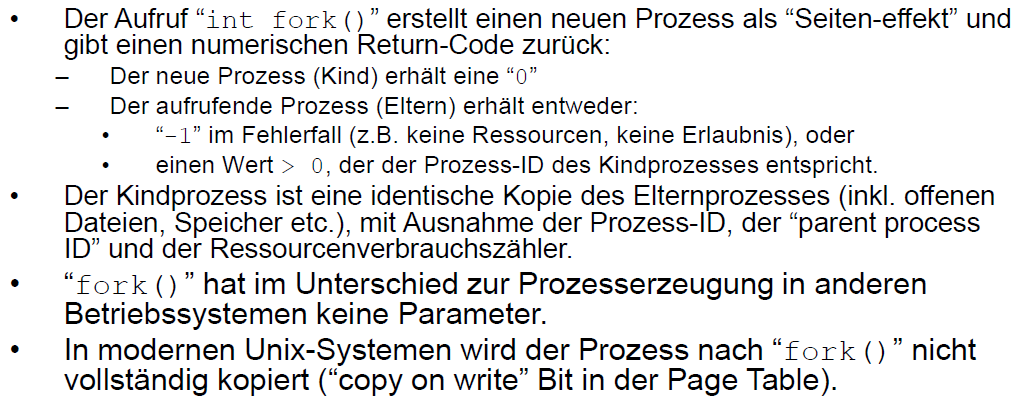
## Prozessbezogene System Calls in Unix

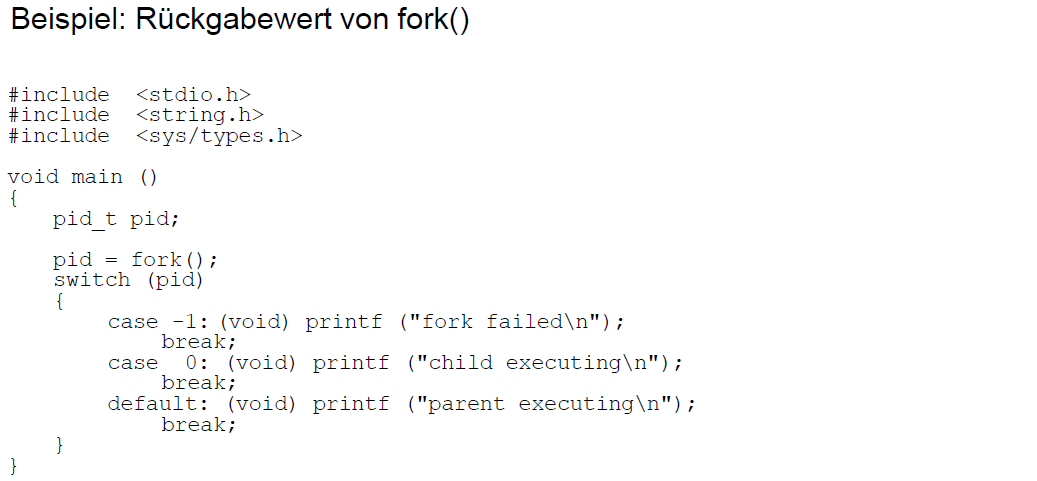


## System Call exec()

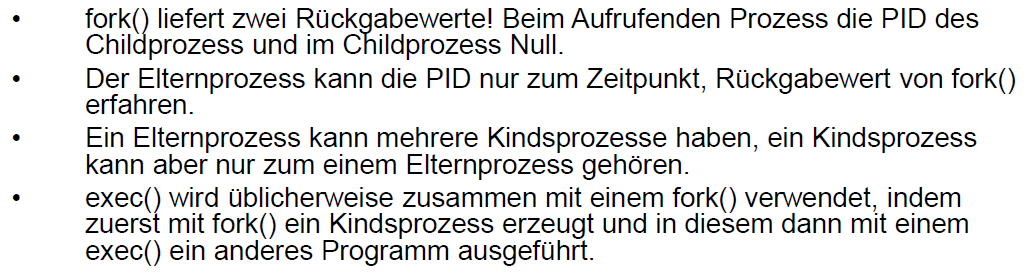


## System Call fork()

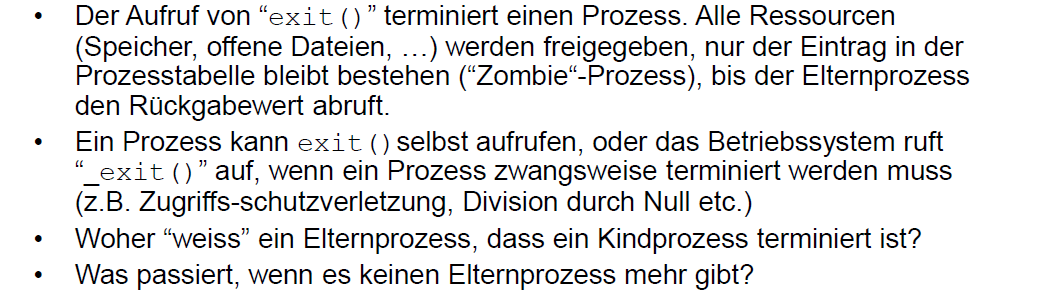




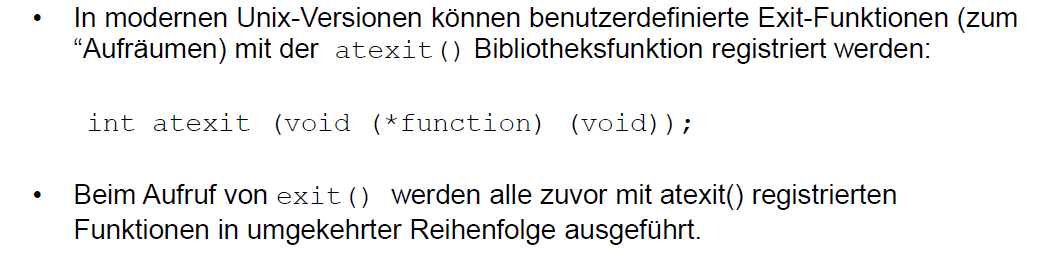
## Vergleich fork() und exec()



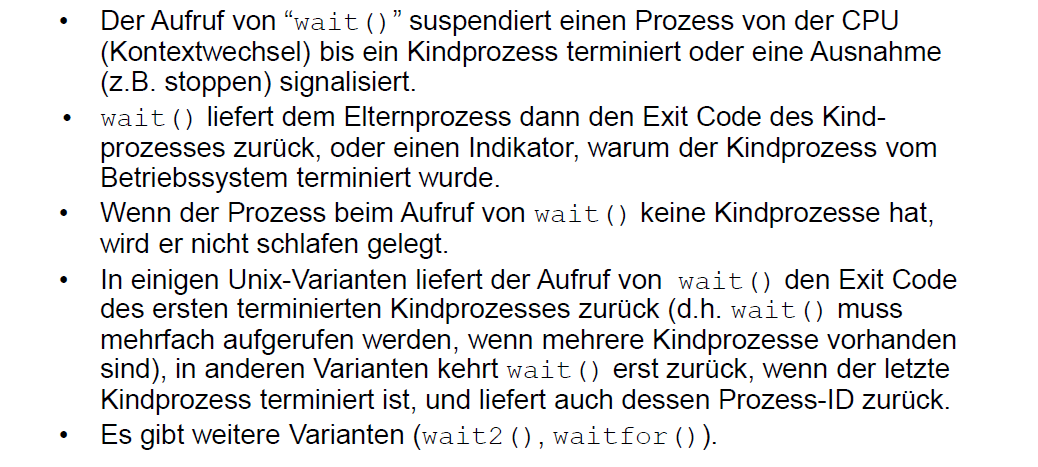
## System Call exit()



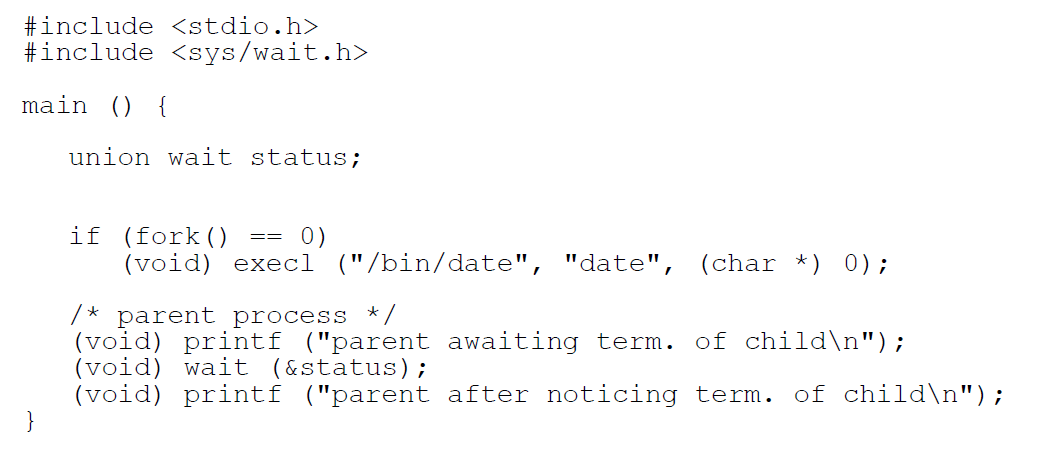
## Funktionen atexit() und on\_exit()



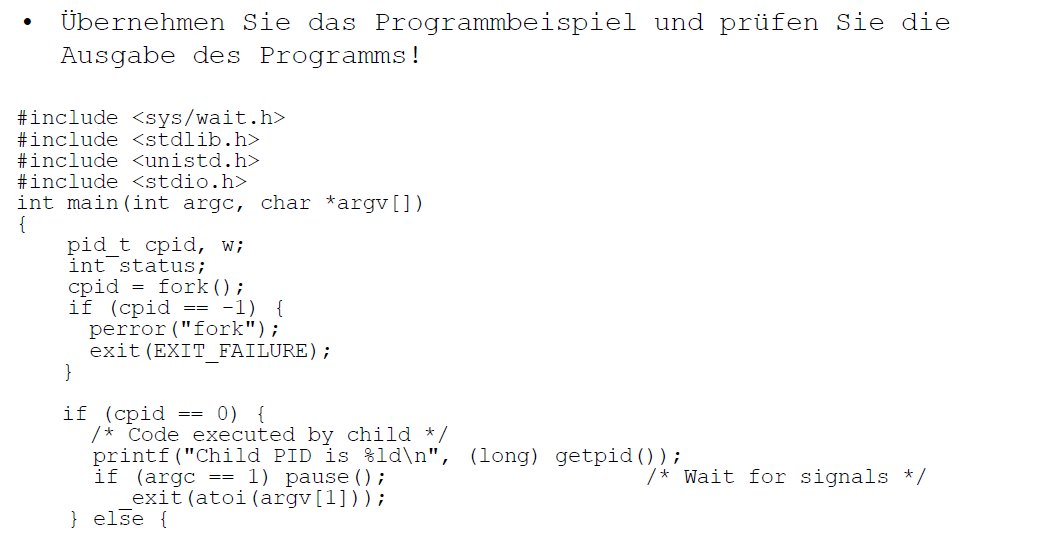
## System Call wait()

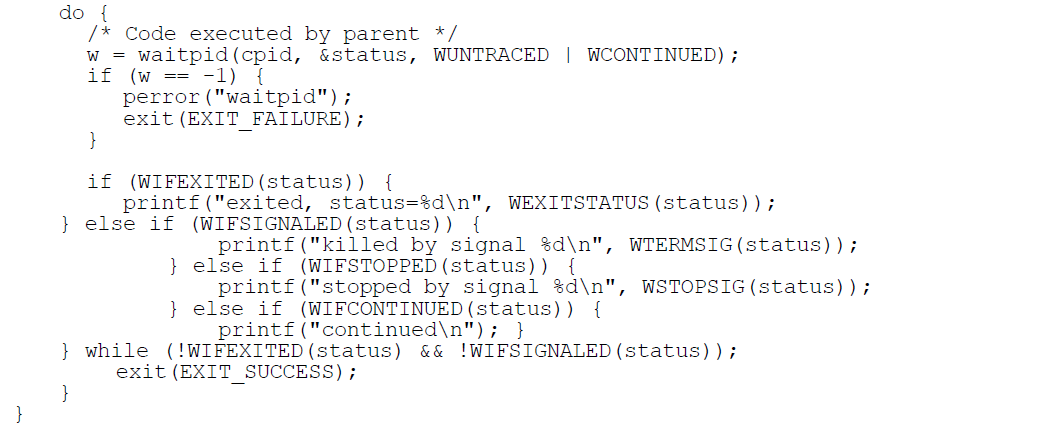


## Programmierung mit wait()

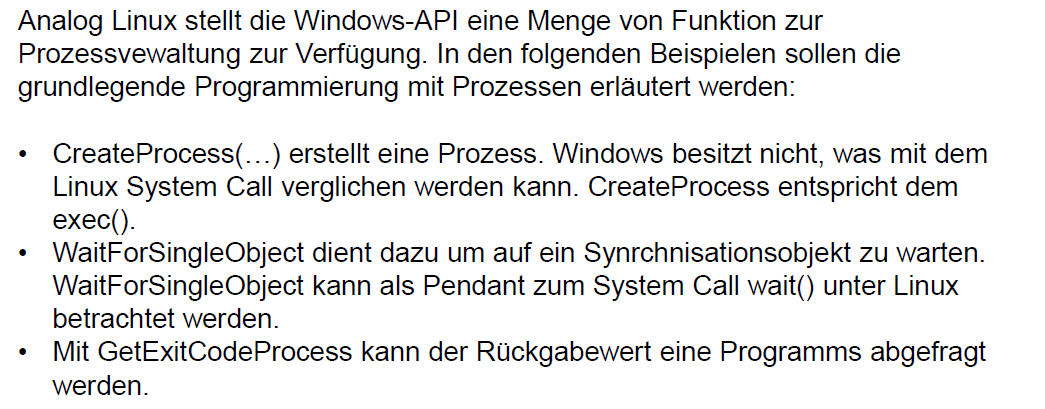


## Auswertung des Exit Code nach wait()

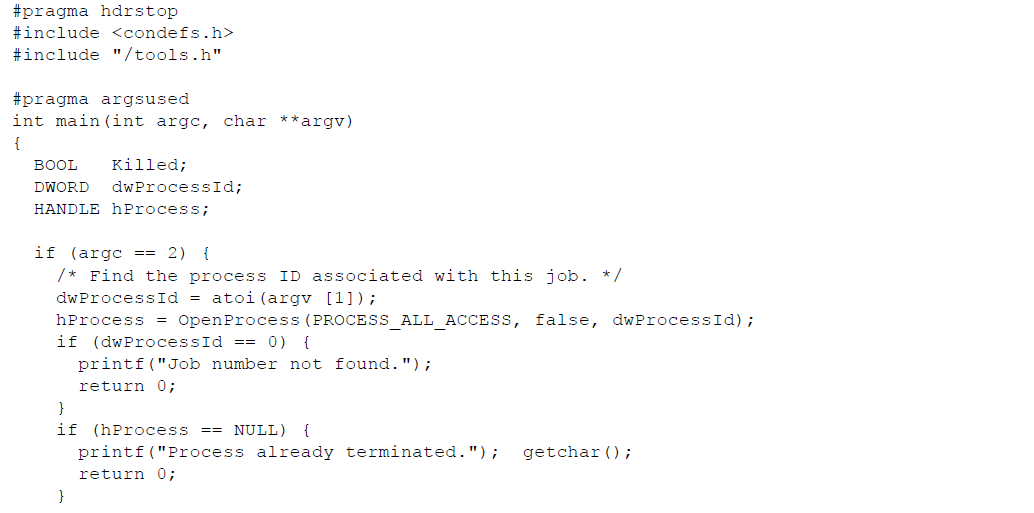


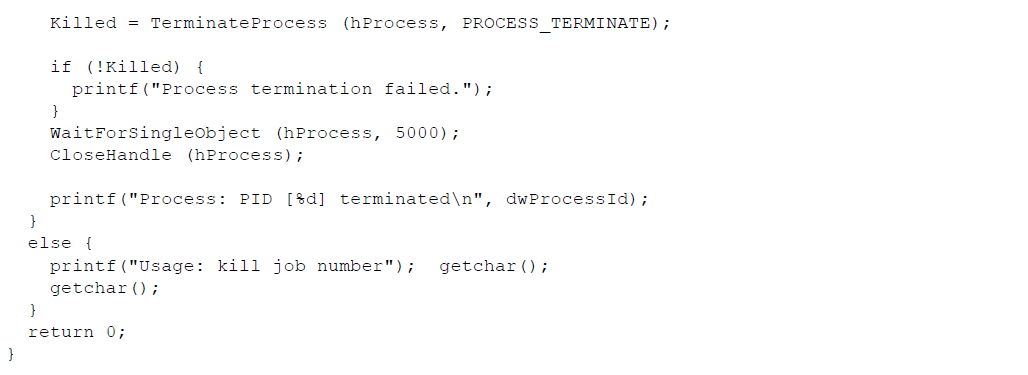


## Prozesse erzeugen mittels Windows API

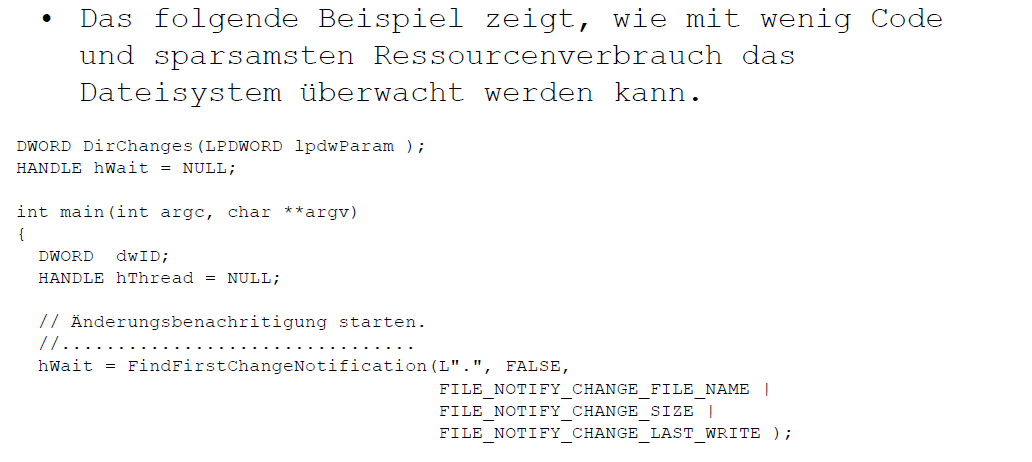


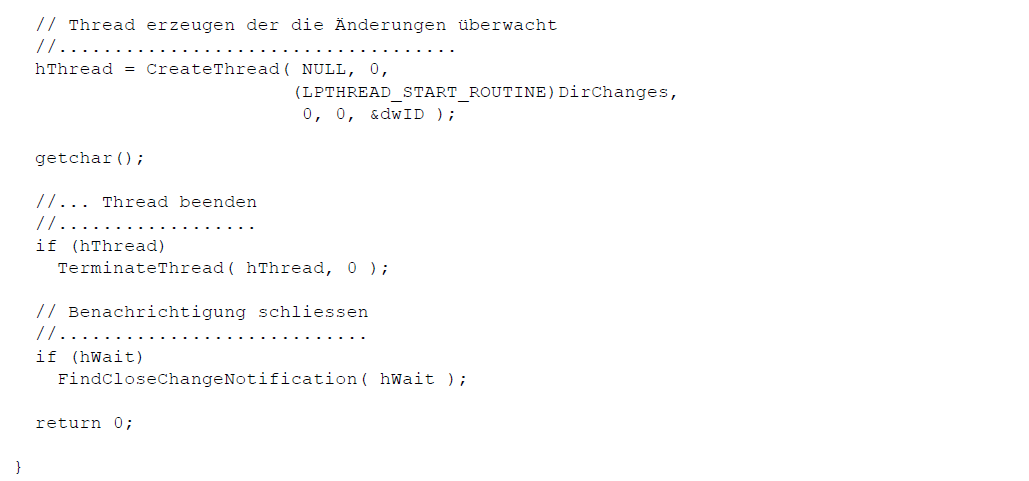
## Beispiel von kill in der Windows API

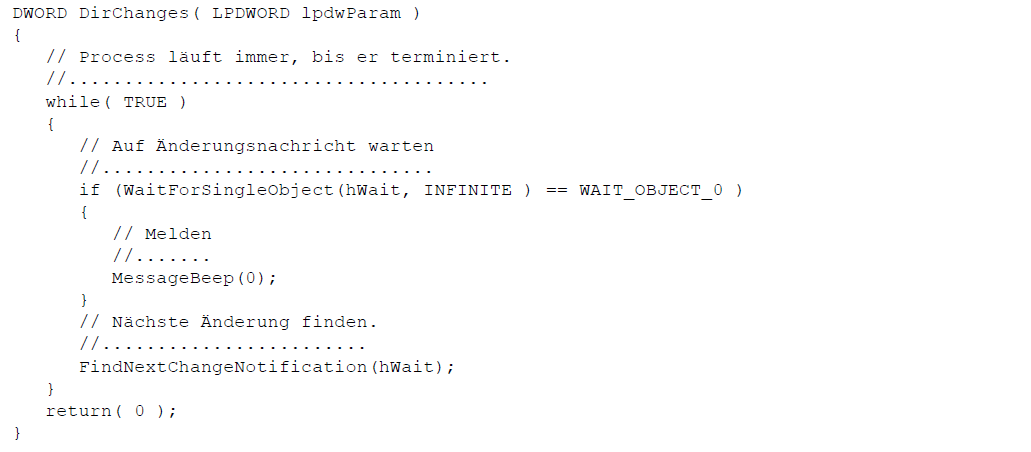




## Beispiel der Dateisystemüberwachung in der Windows API







# Woche 7