FH-OÖ Hagenberg/HSD
Betriebssysteme 3, WS 2025
Übung 7

Name:	**Marco Söllinger**	Aufwand in h:	**3**
Mat.Nr:	**s2410306011**	Punkte:	
Übungsgruppe:	**Gruppe 1**	korrigiert:	

Verzeichnisbaum ausgeben (24 Punkte)

Schreiben Sie ein C-Programm, das für ein angegebenens Verzeichnis als Kommandozeilen-Parameter den vollständigen Verzeichnisbaum mit allen Dateien und Unterverzeichnissen ausgibt. Verwenden Sie zum Durchlaufen des Verzeichnisbaumes die POSIX.1 Funktionen

- `opendir()`
- `readdir()`
- `closedir()`
- `chdir()`
- `getcwd()`

Die Ausgabe ist entsprechend dem folgenden Beispiel zu formatieren:

```
[/home/bes3/workspace/gcc/]
  |---[Math/]
  |   |---[Math]
  |   |---[qt_temp.SZ5707]
  |   |---[qt_temp.Co5707]
  |   |---[Math.c]
  |---[c_test_prog/]
  |   |---[main.c]
  |   |---[Print.h]
  |   |---[Test.h]
  |   |---[main.c~]
  |   |---[Test.c]
  |   |---[Übung2/]
```


1

```
|   |   |---[main.c]
|   |   |---[Print.h]
|   |   |---[Print.c]
|   |---[text.txt]
|   |---[Print.c]
|   |---[Makefile]
|---[TestMake/]
|   |---[test1.c]
|   |---[test2.c~]
|   |---[Makefile_with_dependencies~]
|   |---[Makefile_for_latex_import.txt]
|   |---[Makefile~]
```

Geben Sie bei Aufruf der verschiedenen Systemfunktionen entsprechende Fehlermeldungen aus!

Die maximale Länge eines Pfades ist auf `PATH_MAX` in `limits.h` beschränkt!

Versehen Sie Ihren Quelltext mit entsprechenden Kommentaren. Geben Sie weiters Ihre Ergebnisse und alle Dateien schriftlich und elektronisch ab!

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit Kommentaren! Testen Sie ihre Implementierungen ausführlich! Geben Sie den **Testoutput** mit ab!

Beispiel 1

Es wird keine Lösungsidee gefordert, deshalb wurde der Code entsprechend kommentiert.

Das Programm wurde mit einer Makefile kompiliert.
Fuer das Testen wurde ein testscript erstellt, welches die verschiedenen Befehle testet.

1.1 Code

main.c

```c
#include <dirent.h>
#include <errno.h>
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

static const char *const errUsage = "Usage: %s <directory>\n";
static const char *const errOpenDir = "opendir";
static const char *const errChdirSubdir = "chdir in Unterverzeichnis";
static const char *const errChdirBack = "chdir zurueck";
static const char *const errCloseDir = "closedir";
static const char *const errGetcwdStart = "getcwd (Startverzeichnis)";
static const char *const errChdirTarget = "chdir (Zielverzeichnis)";
static const char *const errGetcwdTarget = "getcwd (Zielverzeichnis)";
static const char *const errChdirBackStart =
    "chdir (zurueck zum Startverzeichnis)";
static const char *const errIsDirCheck = "isDirectory (chdir-Test)";
static const char *const errStartPathTooLong =
    "Fehler: Laenge des Startverzeichnispfads ist groesser als PATH_MAX "
    "(%ld).\n";
static const char *const errTargetPathTooLong =
    "Fehler: Laenge des Zielverzeichnispfads ist groesser als PATH_MAX "
    "(%ld).\n";

typedef enum { ISDIR, NODIR, ERROR } isDir_t;

static isDir_t isDirectory(char const *name) {
  if (chdir(name) == 0) {
    // erfolgreich reingegangen -> also Verzeichnis
    if (chdir("..") == -1) {
      perror(errChdirBack);
      return ERROR;
    }
    return ISDIR;
  }

  if (errno == ENOTDIR) {
    return NODIR;
  }

  return ERROR;
}

// go through directory and print its tree structure
static void print_tree(int depth) {
  DIR *dir;
  struct dirent *entry;

  dir = opendir(".");
  if (dir == NULL) {
    perror(errOpenDir);
    return;
  }

  // go through all elements in directory
  while ((entry = readdir(dir)) != NULL) {
    const char *name = entry->d_name;
```

```
62
63      // skip two directorys that always exist
64      if (strcmp(name, ".") == 0 || strcmp(name, "..") == 0)
65        continue;
66
67      for (int i = 0; i < depth; ++i) {
68        printf("| ");
69      }
70
71      printf("|---[%s", name);
72
73      // checks if element is a directory
74      isDir_t check = isDirectory(name);
75      switch (check) {
76      case NODIR:
77        printf("]\n");
78        break;
79
80      case ISDIR:
81        printf("/]\n");
82
83        if (chdir(name) == -1) {
84          perror(errChdirSubdir);
85          closedir(dir);
86          return;
87        }
88
89        print_tree(depth + 1);
90
91        if (chdir("..") == -1) {
92          perror(errChdirBack);
93          closedir(dir);
94          return;
95        }
96        break;
97
98      case ERROR:
99      default:
100       printf("]\n");
101       perror(errIsDirCheck);
102       closedir(dir);
103       return;
104     }
105   }
106
107   if (closedir(dir) == -1) {
108     perror(errCloseDir);
109   }
110 }
111
112 int main(int argc, char *argv[]) {
113   char startdir[PATH_MAX];
114
115   if (argc != 2) {
116     fprintf(stderr, errUsage, argv[0]);
117     return EXIT_FAILURE;
118   }
119
120   // remember starting directory
121   if (getcwd(startdir, sizeof(startdir)) == NULL) {
122     perror(errGetcwdStart);
123
124     if (errno == ERANGE) {
125       fprintf(stderr, errStartPathTooLong, (long)PATH_MAX);
126     }
127
128     return EXIT_FAILURE;
129   }
130
131   // switch to target directory
132   if (chdir(argv[1]) == -1) {
133     perror(errChdirTarget);
134
135     if (errno == ENOENT) {
136       fprintf(stderr, "Fehler: Pfad '%s' wurde nicht gefunden.\n", argv[1]);
137     } else if (errno == ENOTDIR) {
```

```
138       fprintf(stderr, "Fehler: '%s' ist kein Verzeichnis.\n", argv[1]);
139     } else if (errno == EACCES) {
140       fprintf(stderr, "Fehler: Keine Berechtigung, um '%s' zu betreten.\n",
141               argv[1]);
142     }
143
144     return EXIT_FAILURE;
145   }
146
147   // absolute path of target directory
148   char cwd[PATH_MAX];
149   if (getcwd(cwd, sizeof(cwd)) == NULL) {
150     perror(errGetcwdTarget);
151
152     if (errno == ERANGE) {
153       fprintf(stderr, errTargetPathTooLong, (long)PATH_MAX);
154     }
155
156     // try to switch back to starting directory
157     if (chdir(startdir) == -1) {
158       perror(errChdirBackStart);
159     }
160     return EXIT_FAILURE;
161   }
162
163   // Print first line, the absolute path of the target directory
164   printf("[%s/]\n", cwd);
165
166   // recursively print the directory tree
167   print_tree(0);
168
169   // switch back to starting directory
170   if (chdir(startdir) == -1) {
171     perror(errChdirBackStart);
172     return EXIT_FAILURE;
173   }
174
175   return EXIT_SUCCESS;
176 }
```

1.2 Test

Es wurden folgende Testfälle implementiert und ausgeführt:

testscript.sh

```
1  #!/bin/bash
2
3  rm Verzeichnisbaum
4  make
5
6  EXEC="./Verzeichnisbaum"
7  if [[ ! -x "$EXEC" ]]; then
8    echo "Fehler: $EXEC existiert nicht oder nicht ausfuehrbar"
9    exit 1
10 fi
11
12 echo "==> Starte Testlaufe fuer readDir"
13 echo
14
15 # Hilfsfunktion: Trenner
16 sep() {
17   echo "------------------------------------------------"
18 }
19
20 # Arbeitsverzeichnis vorbereiten
21 TESTDIR="test_readDir_env"
22 rm -rf "$TESTDIR"
23 mkdir "$TESTDIR"
24
25 # ------------------------------------------------------
26 sep
27 echo "TEST 0: Invalid Call (kein Argument)(2 Args)"
28 sep
29 $EXEC
```

```
30 echo
31 $EXEC "Arg1" "Arg2"
32 echo
33
34 # ------------------------------------------------
35 sep
36 echo "TEST 1: Standardverhalten - aktuelles Verzeichnis"
37 sep
38 $EXEC "."
39 echo
40
41 # ------------------------------------------------
42 sep
43 echo "TEST 2: Leeres Verzeichnis"
44 sep
45 mkdir "$TESTDIR/empty"
46 $EXEC "$TESTDIR/empty"
47 echo
48
49 # ------------------------------------------------
50 sep
51 echo "TEST 3: Verzeichnis mit Dateien und Unterverzeichnissen"
52 sep
53 mkdir "$TESTDIR/mixed"
54 touch "$TESTDIR/mixed/a.txt"
55 touch "$TESTDIR/mixed/b.bin"
56 mkdir "$TESTDIR/mixed/subdir"
57 $EXEC "$TESTDIR/mixed"
58 echo
59
60 # ------------------------------------------------
61 sep
62 echo "TEST 4: Nicht-existierendes Verzeichnis"
63 sep
64 $EXEC "$TESTDIR/does_not_exist"
65 echo
66
67 # ------------------------------------------------
68 sep
69 echo "TEST 5: Pfad zeigt auf Datei statt Verzeichnis"
70 sep
71 echo "Hallo Welt" >"$TESTDIR/file.txt"
72 $EXEC "$TESTDIR/file.txt"
73 echo
74
75 # ------------------------------------------------
76 sep
77 echo "TEST 6: Verzeichnis ohne Leserechte (r-Bit fehlt)"
78 sep
79 mkdir "$TESTDIR/noread"
80 touch "$TESTDIR/noread/x"
81 chmod a-r "$TESTDIR/noread"
82
83 $EXEC "$TESTDIR/noread"
84
85 chmod u+r "$TESTDIR/noread"
86 echo
87
88 # ------------------------------------------------
89 sep
90 echo "TEST 7: Verzeichnis ohne Execute-Rechte (x-Bit fehlt)"
91 sep
92 mkdir "$TESTDIR/nosearch"
93 touch "$TESTDIR/nosearch/z"
94 chmod a-x "$TESTDIR/nosearch"
95
96 $EXEC "$TESTDIR/nosearch"
97
98 # Rechte wiederherstellen
99 chmod u+x "$TESTDIR/nosearch"
100 echo
101
102 # ------------------------------------------------
103 sep
104 echo "TEST 8: Sehr langer Pfad "
105 sep
```

```
106
107 # Hole PATH_MAX vom System (typisch 4096, aber wir fragen lieber)
108 PATH_LIMIT=$(getconf PATH_MAX . 2>/dev/null)
109 if [[ -z "$PATH_LIMIT" ]]; then
110   PATH_LIMIT=4096 # fallback, falls getconf scheitert
111 fi
112
113 LONGBASE="$TESTDIR/longpath"
114 rm -rf "$LONGBASE"
115 mkdir -p "$LONGBASE"
116
117 cur="$LONGBASE"
118 segment="subdir_xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
119
120 while :; do
121   new="$cur/$segment"
122   if (${#new} >= PATH_LIMIT + 10)); then
123     break
124   fi
125   mkdir -p "$new" || break
126   cur="$new"
127 done
128
129 echo "Laenge des erzeugten Pfads: ${#cur} (PATH_MAX ~ $PATH_LIMIT)"
130 echo "Aufruf: $EXEC \"$cur\""
131 $EXEC "$cur"
132 echo
133
134 # -----------------------------------------------------
135 sep
136 echo "TESTS FERTIG"
137 sep
138
139 exit 0
```

Terminal Output

```
1  flashfish@fedora ~/D/R/F/B/Uebung07 (main)> ./testscript.sh
2  clang  -Wall -Wextra -c main.c -o build/main.o
3  clang -g -o Verzeichnisbaum build/main.o -lm
4  ==> Starte Testlaufe fuer readDir
5
6  -------------------------------------------
7  TEST 0: Invalid Call (kein Argument)(2 Args)
8  -------------------------------------------
9  Usage: ./Verzeichnisbaum <directory>
10
11 Usage: ./Verzeichnisbaum <directory>
12
13 -------------------------------------------
14 TEST 1: Standardverhalten - aktuelles Verzeichnis
15 -------------------------------------------
16 [/home/flashfish/Documents/Repo/FH/BSY3/Uebung07/]
17 |---[latex/]
18 | |---[Uebung7.pdf]
19 | |---[ue9.aux]
20 | |---[ue9.fdb_latexmk]
21 | |---[ue9.log]
22 | |---[ue9.pdf]
23 | |---[ue9.tex]
24 | |---[ue9.fls]
25 | |---[ue9.synctex.gz]
26 |---[Makefile]
27 |---[build/]
28 | |---[main.o]
29 |---[main.c]
30 |---[testscript.sh]
31 |---[Verzeichnisbaum]
32 |---[test_readDir_env/]
33
34 -------------------------------------------
35 TEST 2: Leeres Verzeichnis
36 -------------------------------------------
37 [/home/flashfish/Documents/Repo/FH/BSY3/Uebung07/test_readDir_env/empty/]
38
39 -------------------------------------------
```

```
TEST 3: Verzeichnis mit Dateien und Unterverzeichnissen
-----------------------------------------------
[/home/flashfish/Documents/Repo/FH/BSY3/Uebung07/test_readDir_env/mixed/]
|---[a.txt]
|---[b.bin]
|---[subdir/]

-----------------------------------------------
TEST 4: Nicht-existierendes Verzeichnis
-----------------------------------------------
chdir (Zielverzeichnis): No such file or directory
Fehler: Pfad 'test_readDir_env/does_not_exist' wurde nicht gefunden.

-----------------------------------------------
TEST 5: Pfad zeigt auf Datei statt Verzeichnis
-----------------------------------------------
chdir (Zielverzeichnis): Not a directory
Fehler: 'test_readDir_env/file.txt' ist kein Verzeichnis.

-----------------------------------------------
TEST 6: Verzeichnis ohne Leserechte (r-Bit fehlt)
-----------------------------------------------
[/home/flashfish/Documents/Repo/FH/BSY3/Uebung07/test_readDir_env/noread/]
opendir: Permission denied

-----------------------------------------------
TEST 7: Verzeichnis ohne Execute-Rechte (x-Bit fehlt)
-----------------------------------------------
chdir (Zielverzeichnis): Permission denied
Fehler: Keine Berechtigung, um 'test_readDir_env/nosearch' zu betreten.

-----------------------------------------------
TEST 8: Sehr langer Pfad
-----------------------------------------------
Laenge des erzeugten Pfads: 4095 (PATH_MAX ~ 4096)
Aufruf: ./Verzeichnisbaum "test_readDir_env/longpath/subdir_xxxxxxxxxxxxxxxxxxxxxxxxxxxxx/ ...
     {I removed the rest of the path so that it fits here} ... /
    subdir_xxxxxxxxxxxxxxxxxxxxxxxxxxxx/"
getcwd (Zielverzeichnis): Numerical result out of range
Fehler: Laenge des Zielverzeichnispfads ist groesser als PATH_MAX (4096).

-----------------------------------------------
TESTS FERTIG
-----------------------------------------------
```