SWO 3x Übung zu Softwareentwicklung mit klassischen Sprachen u. Bibliotheken 3 **WS 2014/15, Übung 04**

Abgabetermin: Sa in der KW 46

☐ **Gr. 1,** DI Franz Gruber-Leitner **Name** Roman Lumetsberger **Aufwand in h** 6

☒ **Gr. 2,** Dr. Erik Pitzer

Punkte _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

1. Stdlib & Find (8 + 4 + 6 + 6 Punkte)

Die Standardbibliothek stellt sehr viel an Funktionalität bereits zur Verfügung. Um sie effizient zu verwenden, muss man die vorhandene Dokumentation aber sorgfältig studieren. Um das zu üben sollen Sie in diesem Beispiel eine einfache Version des UNIX Werkzeugs `find` implementieren, das einen Verzeichnisbaum rekursiv durchläuft und Dateien mit bestimmten Eigenschaften sucht, bzw. verarbeitet, wie Sie in der *manpage* von find nachlesen können.

(a) Implementieren Sie dazu im ersten Schritt mit Hilfe der Standardbibliothek eine Funktion, die, ausgehend von einem Startverzeichnis, rekursiv alle Verzeichnisse und Dateien durchläuft und für alle regulären Dateien mit einer beliebigen Funktion verarbeitet, die als Funktionszeiger übergeben wird, also z.B. folgende Schnittstelle erfüllt:

```
typedef void (*Visitor)(char *pathname, struct stat *stat);
void walkDir(char *dirname, Visitor visitor);
```

Die übergebene Funktion `visitor`, erhält also für jede Datei, den vollständigen Dateinamen, sowie die Dateiattribute (`man lstat`).

(b) Die jeweilige Visitor-Funktion soll per Kommandozeilenargument ausgewählt werden können. Es soll dabei berücksichtigt werden, dass manche dieser Visitor-Funktionen ein zusätzliches Argumente enthalten können. Es kann angenommen werden, dass alle möglichen Visitor-Funktionen bereits bekannt sind und somit hartcodiert werden können.

(c) Implementieren Sie eine erste Visitor-Funktion, die einfach alle Dateien und deren wichtigste Attribute ausgibt. Es soll mindestens folgendes ausgegeben werden:

- vollständiger Dateiname mit Pfad
- letztes Änderungsdatum
- Berechtigungen
- Größe

Eine mögliche Ausgabe könnte z.B. so aussehen:

```
user@ubuntu:~/swo3/homeworks/u4$ find .. –print
../u1/Makefile~           rwxrwxrwx      136 09/15/14 14:21
../u1/prime               rwxrwxrwx     7581 09/18/14 14:03
../u1/prime.c             rwxrwxrwx      951 09/18/14 14:03
../u1/prime.c~            rwxrwxrwx      949 09/15/14 14:09
../u1/triangle            rwxrwxrwx     7493 09/18/14 14:03
../u1/triangle.c          rwxrwxrwx      733 09/18/14 14:03
../u1/triangle.c~         rwxrwxrwx      732 09/15/14 14:21
                             .
                             .
                             .
../u4/find                rwxrwxrwx    21743 10/20/14 15:59
../u4/find.c              rwxrwxrwx     4635 10/20/14 15:59
../u4/find.c~             rwxrwxrwx     4620 10/20/14 15:59
../u4/find.o              rwxrwxrwx    17552 10/20/14 15:59
../u4/Makefile            rwxrwxrwx      108 10/20/14 13:10
../u4/Makefile~           rwxrwxrwx      109 10/20/14 13:10
```

(d) Implementieren Sie eine zweite Visitor-Funktion, die alle Dateien nach einer bestimmten Zeichenkette durchsucht und nur passende Zeilen ausgibt, z.B:

```
user@ubuntu:~/swo3/homeworks/u4$ find . –grep failed
find.c:115:21 "    printf("failed to open directory \"%s\"\n", dirname);"
find.c:123:25 "        printf("failed to stat \"%s\"\n", filename);"
find.c~:115:21 "    printf("failed to open directory \"%s\"\n", dirname);"
find.c~:123:25 "        printf("failed to stat \"%s\"\n", filename);"
```

Hinweis: Um Sie nicht beim Aufstöbern der Dokumentation verzweifeln zu lassen, hier noch eine Liste mit potentiell nützlichen Funktionen:

- `dirent(), readdir()` um Verzeichnisse zu traversieren
- `stat(), S_ISDIR()` um Datei- und Verzeichniseigenschaften abzufragen
- `localtime(), strftime()` um das Datum zu formatieren
- `fopen(), getline()` um Dateien zeilenweise zu lesen
- `strstr(), strlen(), strcpy(), strcat(), strcmp()` Funktionen von Zeichenketten

Außerdem hilfreich könnte das Studium den *manpage* über *man* selbst hilfreich sein (`$ man man`), sowie das Kommando `apropos` zum Finden von relevanten *manpages*.

1 Aufgabe 1 - Stdlib / Find

1.1 Lösungsidee

Es müssen alle Dateien und Verzeichnisse rekursiv durchlaufen werden und für jede reguläre Datei muss ein Funktionszeiger aufgerufen werden. Dieser implementiert dann eine spezielle Funktion.

1.1.1 a) walkDir

Die erste Aufgabe implementiert die rekursive Funktion, um alle Dateien zu finden und den Funktionszeiger aufzurufen. Diese läuft über alle Dateien und Verzeichnisse in dem übergebenen Pfad.

- Wird ein Verzeichnis gefunden, wird die Funktion wieder rekursiv mit dem neuen Verzeichnis aufgerufen.
- Wird eine **reguläre** Datei gefunden, dann werden die Dateistati ausgelesen und der Funktionszeiger aufgerufen.

Folgende Funktionen der Standardbibliothek werden benötigt:

- opendir/closedir: Öffnet/Schließt ein Verzeichnis um alle Dateien/Ordner auszulesen.
  Liefert einen Zeiger auf das Verzeichnis, oder NULL, wenn ein Fehler aufgetreten ist.

```
DIR *opendir(const char *name);
int closedir(DIR *dirp);
```

- readdir: Liefert den nächsten Verzeichniseintrag eines mit opendir geöffneten Verzeichnisses.
  Liefert *NULL*, wenn das Ende erreicht wurde oder ein Fehler aufgetreten ist.

```
struct dirent *readdir(DIR *dirp);
```

- lstat: Liefert die Dateistati für die angegebe Datei.
  Liefert *0* bei Erfolg, *-1* bei einem Fehler.
  buf enthält die Dateistati, die dann mit den definierten Makros *S_** ausgewertet werden können.

```
/* needs _XOPEN_SOURCE >= 700 */
int lstat(const char *path, struct stat *buf);
```

1.1.2 b) Auswahl der Funktion

Im Hauptprogramm müssen die Parameter ausgewertet werden, um die korrekte Funktion festlegen zu können.
Die Funktion walkDir muss um optionale Parameter $StdArg$ erweitert werden.
Anmerkung: Bei der Umsetztung wurde auf eine möglichst generische Schnittstelle geachtet, die

es erlaubt, neue Visitors zu implementieren, ohne die Methode *walkDir* ändern zu müssen.
Die Übergabe der generischen Parameter wird mit dem Datentyp *va_list* implementiert, somit können beliebige Datentypen and die Visitor Funktion übergeben werden. **Folgende Funktionen der Standardbibliothek werden benötigt:**

- va_start: Initialisiert die Liste mit den optionalen Parametern. (Makro)

```
void va_start(va_list ap, last);
```

- va_end: Gibt die Liste mit den optionalen Parametern wieder frei.

```
void va_end(va_list ap);
```

- va_arg: Gibt den nächsten Parameter als *type* zurück.
  Achtung: Der Datentyp wird nicht konvertiert und muss genau übereinstimmen, sonst ist das Ergebnis undefiniert.

```
type va_arg(va_list ap, type);
```

1.1.3 c) print

Hier muss eine neue Funktion, die dem Funktionsmuster *Visitor* folgt, implementiert werden.
Diese wertet die Dateistati aus und gibt sie formatiert aus.
Die Zugriffsrechte werden über den Wert *st_mode* ausgelesen.
Die Zeit wird über den Wert st_mtime ausgelesen.

Wichtig: Das im Code vorhandene **#Pragma** wird bewusst verwendet, um keine Warnings beim kompilieren zu bekommen, da nicht alle Parameter verwendet werden.

Folgende Funktionen der Standardbibliothek werden benötigt:

- localtime: Wandelt die übergebene Zeit in die Datenstruktur *struct tm* um und konvertiert sie dabei in die aktuelle Zeitzone des Users.

```
struct tm *localtime(const time_t *timep);
```

- strftime: Formatiert die angegebene Zeit und liefert sie als Zeichenkette zurück.
  Für die Formatierung wird hier **%c** verwendet. Dies formatiert das Datum/Zeit in das im Benutzerprofil definierten Format.

```
size_t strftime(char *s, size_t max, const char *format,
                const struct tm *tm);
```

1.1.4 d) grep

Hier muss eine neue Funktion, die dem Funktionsmuster *Visitor* folgt, implementiert werden. Diese benötigt einen weiteren Parameter, der über das StdArg Interface abgerufen werden kann.

Die Funktion öffnet dann die Datei, ließt sie zeilenweise aus und sucht nach der übergebenen Zeichenkette.
Wird die Zeichenkette gefunden, wird die Zeile ausgegeben.

Wichtig: Das im Code vorhandene **#Pragma** wird bewusst verwendet, um keine Warnings beim kompilieren zu bekommen, da nicht alle Parameter verwendet werden.

Folgende Funktionen der Standardbibliothek werden benötigt:

- fopen/fclose: Öffnet/Schließt eine Datei. Dabei kann mit dem Parameter *mode* angegeben werden, ob sie zum Lesen oder Schreiben geöffnet werden soll.
  Die Funktion liefert einen Zeiger auf die Datei, oder NULL, wenn sie nicht geöffnet werden konnte.

```
FILE *fopen(const char *path, const char *mode);
      int fclose(FILE *fp);
```

- getline: Liest eine Zeile einer Datei aus und gibt diese im Parameter **lineptr** zurück.
  Dabei wird der benötigte Speicherplatz für die Zeichenkette automatisch allokiert, sofern **lineptr* NULL ist.
  Wichtig: Der allokierte Speicherplatz muss wieder freigegeben werden.
  Die Funktion liefert *-1*, wenn das Ende der Datei erreicht ist oder ein Fehler auftritt.

```
/* needs _XOPEN_SOURCE >= 500 */
ssize_t getline(char **lineptr, size_t *n, FILE *stream);
```

1.2 Sourcecode

```c
/* used for getline and lstat */
#define _XOPEN_SOURCE 700

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <sys/stat.h>

/* definition of the function pointer used for walkDir */
typedef void (*Visitor) (char *pathname, struct stat* stat, va_list args);

/* walks recursive across the given path and calls the visitor function
   the args parameter is used for giving the visitor function more arguments */
void walkDirWithArguments(char *dirname, Visitor visitor, va_list args) {
  DIR * directory;
  struct dirent * entry;
  struct stat entryStat;
  char *pathname;

  directory = opendir(dirname);
  if(directory == NULL) {
    printf("failed to open directory %s\n", dirname);
    return;
  }

  entry = readdir(directory);
  while (entry != NULL) {
    /* skip . and .. entries */
    if ((strcmp(entry->d_name, ".") != 0) && (strcmp(entry->d_name, "..") != 0)) {

      /* malloc space for the full path (dir + name + / + \0) */
      pathname = (char *) malloc(sizeof(char) * (strlen(dirname) + strlen(entry->d_name) + 2));
      if (pathname == NULL) {
        printf("out of memory\n");
        abort();
      }

      /* create full path */
      sprintf(pathname,"%s/%s",dirname,entry->d_name);

      if(lstat(pathname, &entryStat) == 0) {
        /* check if entry is a directory */
        if (S_ISDIR(entryStat.st_mode)) {
```

```
47            walkDirWithArguments(pathname, visitor, args);
48          }
49          /* check if entry is a regular file */
50          else if (S_ISREG(entryStat.st_mode)) {
51            visitor(pathname, &entryStat, args);
52          }
53        }
54        else {
55          printf("failed to stat %s\n", pathname);
56        }
57
58        free(pathname);
59
60      }
61      entry = readdir(directory);
62    }
63
64    closedir(directory);
65  }
66
67  /* wrapper function used for giving the variable arguments to the real function */
68  void walkDir(char *dirname, Visitor visitor, ...) {
69    va_list list;
70
71    /* get the argument list */
72    va_start(list, visitor);
73    walkDirWithArguments(dirname, visitor, list);
74    /* free the arguments */
75    va_end(list);
76  }
77
78
79  /* disable unsed parameter warning because of the generic interface */
80  #pragma GCC diagnostic push
81  #pragma GCC diagnostic ignored "-Wunused-parameter"
82
83  /* visitor function used for showing the filename and details */
84  void print(char * path, struct stat* stat, va_list args ) {
85    char timeBuffer[80];
86    struct tm *locTime;
87
88    printf( (stat->st_mode & S_IRUSR) ? "r" : "-");
89    printf( (stat->st_mode & S_IWUSR) ? "w" : "-");
90    printf( (stat->st_mode & S_IXUSR) ? "x" : "-");
91    printf( (stat->st_mode & S_IRGRP) ? "r" : "-");
92    printf( (stat->st_mode & S_IWGRP) ? "w" : "-");
93    printf( (stat->st_mode & S_IXGRP) ? "x" : "-");
94    printf( (stat->st_mode & S_IROTH) ? "r" : "-");
95    printf( (stat->st_mode & S_IWOTH) ? "w" : "-");
```

```
96    printf( (stat->st_mode & S_IXOTH) ? "x" : "-");
97    printf("\t");
98    printf("%d\t",(int) stat->st_size);
99
100   locTime = localtime(&stat->st_mtime);
101   /* format time */
102   strftime(timeBuffer, sizeof(timeBuffer), "%c", locTime);
103
104   printf("%s\t",timeBuffer);
105   printf("%s\n",path);
106
107 }
108 /* enable warnings again */
109 #pragma GCC diagnostic  pop
110
111
112 /* disable unsed parameter warning because of the generic interface */
113 #pragma GCC diagnostic push
114 #pragma GCC diagnostic ignored "-Wunused-parameter"
115
116 /* read the file and prints all occurances of the given string
117    the string has to be given as parameter type char * */
118 void grep(char * path, struct stat* stat, va_list args) {
119   FILE *fp;
120   char *searchString;
121   char *line = NULL;
122   char *strPosition;
123   size_t len = 0;
124   ssize_t read;
125   int linenum;
126
127   searchString = va_arg(args, char *);
128
129   fp = fopen(path, "r");
130   if (fp == NULL) {
131     printf("could not open file %s", path);
132     return;
133   }
134
135   linenum = 1;
136   read = getline(&line, &len, fp);
137   while (read != -1) {
138     strPosition = strstr(line, searchString);
139     if( strPosition != NULL ){
140       printf("%s:%d:%d %s", path, linenum, (strPosition - line), line);
141     }
142     linenum++;
143     read = getline(&line, &len, fp);
144   }
```

```
145
146   if (line != NULL) {
147     free(line);
148   }
149
150   fclose(fp);
151 }
152 /* enable warnings again */
153 #pragma GCC diagnostic  pop
154
155
156 int main(int argc, char *argv[])
157 {
158   if (argc < 3) {
159     printf("Invalid parameter\n");
160     printf("Usage: %s dir function\n", argv[0]);
161     printf("\tfunction -print: shows the files and details\n");
162     printf("\tfunction -grep content: shows the files with the given content\n");
163     return EXIT_SUCCESS;
164   }
165
166   if (strcmp(argv[2], "-print") == 0) {
167     walkDir(argv[1], &print);
168   }
169   else if(strcmp(argv[2], "-grep") == 0) {
170     if (argc < 4) {
171         printf("Invalid parameter\n");
172         printf("You have to give a search string\n");
173         return EXIT_SUCCESS;
174     }
175     walkDir(argv[1], &grep, argv[3]);
176   }
177   else {
178     printf("invalid function\n");
179   }
180
181   return EXIT_SUCCESS;
182 }
```

1.3 Testfälle

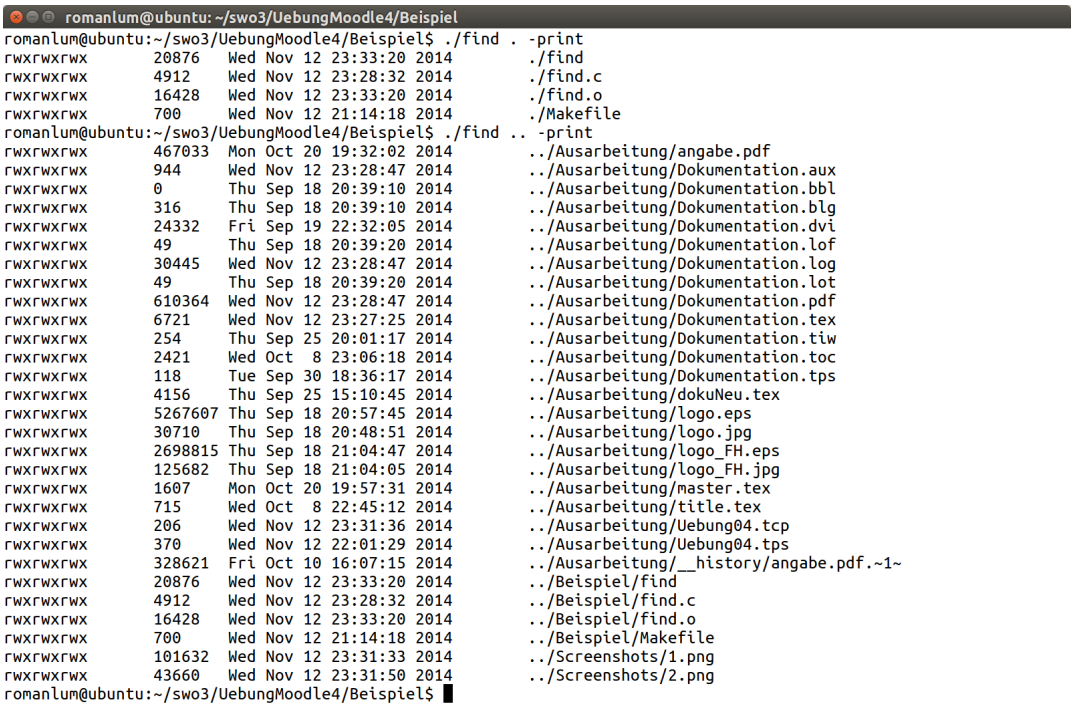
1.3.1 Testfall 1 - Ungültige Parameter

```
romanlum@ubuntu: ~/swo3/UebungMoodle4
romanlum@ubuntu:~/swo3/UebungMoodle4$ ./find
Invalid parameter
Usage: ./find dir function
        function -print: shows the files and details
        function -grep content: shows the files with the given content
romanlum@ubuntu:~/swo3/UebungMoodle4$ ./find -p
Invalid parameter
Usage: ./find dir function
        function -print: shows the files and details
        function -grep content: shows the files with the given content
romanlum@ubuntu:~/swo3/UebungMoodle4$ ./find . -grep
Invalid parameter
You have to give a search string
romanlum@ubuntu:~/swo3/UebungMoodle4$
```

1.3.2 Testfall 2 - Ungültiges Verzeichnis

```
romanlum@ubuntu: ~/swo3/UebungMoodle4
romanlum@ubuntu:~/swo3/UebungMoodle4$ ./find alsdkfj -print
failed to open directory alsdkfj
romanlum@ubuntu:~/swo3/UebungMoodle4$
```

1.3.3 Testfall 3 - Funktion print



1.3.4 Testfall 4 - Valgrind Funktion print

Prüft, ob der allokierte Speicher wieder freigegeben wurde.

```
romanlum@ubuntu: ~/swo3/UebungMoodle4/Beispiel

romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$ valgrind ./find .. -print
==5965== Memcheck, a memory error detector
==5965== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==5965== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==5965== Command: ./find .. -print
==5965==
rwxrwxrwx       467033  Mon Oct 20 19:32:02 2014        ../Ausarbeitung/angabe.pdf
rwxrwxrwx       944     Wed Nov 12 23:28:47 2014        ../Ausarbeitung/Dokumentation.aux
rwxrwxrwx       0       Thu Sep 18 20:39:10 2014        ../Ausarbeitung/Dokumentation.bbl
rwxrwxrwx       316     Thu Sep 18 20:39:10 2014        ../Ausarbeitung/Dokumentation.blg
rwxrwxrwx       24332   Fri Sep 19 22:32:05 2014        ../Ausarbeitung/Dokumentation.dvi
rwxrwxrwx       49      Thu Sep 18 20:39:20 2014        ../Ausarbeitung/Dokumentation.lof
rwxrwxrwx       30445   Wed Nov 12 23:28:47 2014        ../Ausarbeitung/Dokumentation.log
rwxrwxrwx       49      Thu Sep 18 20:39:20 2014        ../Ausarbeitung/Dokumentation.lot
rwxrwxrwx       610364  Wed Nov 12 23:28:47 2014        ../Ausarbeitung/Dokumentation.pdf
rwxrwxrwx       6721    Wed Nov 12 23:27:25 2014        ../Ausarbeitung/Dokumentation.tex
rwxrwxrwx       254     Thu Sep 25 20:01:17 2014        ../Ausarbeitung/Dokumentation.tiw
rwxrwxrwx       2421    Wed Oct  8 23:06:18 2014        ../Ausarbeitung/Dokumentation.toc
rwxrwxrwx       118     Tue Sep 30 18:36:17 2014        ../Ausarbeitung/Dokumentation.tps
rwxrwxrwx       4156    Thu Sep 25 15:10:45 2014        ../Ausarbeitung/dokuNeu.tex
rwxrwxrwx       5267607 Thu Sep 18 20:57:45 2014        ../Ausarbeitung/logo.eps
rwxrwxrwx       30710   Thu Sep 18 20:48:51 2014        ../Ausarbeitung/logo.jpg
rwxrwxrwx       2698815 Thu Sep 18 21:04:47 2014        ../Ausarbeitung/logo_FH.eps
rwxrwxrwx       125682  Thu Sep 18 21:04:05 2014        ../Ausarbeitung/logo_FH.jpg
rwxrwxrwx       1607    Mon Oct 20 19:57:31 2014        ../Ausarbeitung/master.tex
rwxrwxrwx       715     Wed Oct  8 22:45:12 2014        ../Ausarbeitung/title.tex
rwxrwxrwx       206     Wed Nov 12 23:31:36 2014        ../Ausarbeitung/Uebung04.tcp
rwxrwxrwx       370     Wed Nov 12 22:01:29 2014        ../Ausarbeitung/Uebung04.tps
rwxrwxrwx       328621  Fri Oct 10 16:07:15 2014        ../Ausarbeitung/__history/angabe.pdf.~1~
rwxrwxrwx       20876   Wed Nov 12 23:33:20 2014        ../Beispiel/find
rwxrwxrwx       4912    Wed Nov 12 23:28:32 2014        ../Beispiel/find.c
rwxrwxrwx       16428   Wed Nov 12 23:33:20 2014        ../Beispiel/find.o
rwxrwxrwx       700     Wed Nov 12 21:14:18 2014        ../Beispiel/Makefile
rwxrwxrwx       101632  Wed Nov 12 23:31:33 2014        ../Screenshots/1.png
rwxrwxrwx       43660   Wed Nov 12 23:31:50 2014        ../Screenshots/2.png
rwxrwxrwx       397954  Wed Nov 12 23:33:47 2014        ../Screenshots/3.png
rwxrwxrwx       51750   Wed Nov 12 23:34:07 2014        ../Screenshots/4.png
==5965==
==5965== HEAP SUMMARY:
==5965==     in use at exit: 0 bytes in 0 blocks
==5965==   total heap usage: 77 allocs, 77 frees, 166,620 bytes allocated
==5965==
==5965== All heap blocks were freed -- no leaks are possible
==5965==
==5965== For counts of detected and suppressed errors, rerun with: -v
==5965== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$
```

1.3.5 Testfall 5 - Funktion grep

```
romanlum@ubuntu: ~/swo3/UebungMoodle4/Beispiel
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$ ./find .. -grep failed
../Beispiel/find.c:25:12            printf("failed to open directory %s\n", dirname);
../Beispiel/find.c:55:16                printf("failed to stat %s\n", pathname);
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$
```

1.3.6 Testfall 6 - Funktion grep (mehr Fundstellen)

```
romanlum@ubuntu: ~/swo3/UebungMoodle4/Beispiel
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$ ./find .. -grep printf
../Beispiel/find.c:25:4     printf("failed to open directory %s\n", dirname);
../Beispiel/find.c:37:8         printf("out of memory\n");
../Beispiel/find.c:42:7       sprintf(pathname,"%s/%s",dirname,entry->d_name);
../Beispiel/find.c:55:8         printf("failed to stat %s\n", pathname);
../Beispiel/find.c:88:2   printf( (stat->st_mode & S_IRUSR) ? "r" : "-");
../Beispiel/find.c:89:2   printf( (stat->st_mode & S_IWUSR) ? "w" : "-");
../Beispiel/find.c:90:2   printf( (stat->st_mode & S_IXUSR) ? "x" : "-");
../Beispiel/find.c:91:2   printf( (stat->st_mode & S_IRGRP) ? "r" : "-");
../Beispiel/find.c:92:2   printf( (stat->st_mode & S_IWGRP) ? "w" : "-");
../Beispiel/find.c:93:2   printf( (stat->st_mode & S_IXGRP) ? "x" : "-");
../Beispiel/find.c:94:2   printf( (stat->st_mode & S_IROTH) ? "r" : "-");
../Beispiel/find.c:95:2   printf( (stat->st_mode & S_IWOTH) ? "w" : "-");
../Beispiel/find.c:96:2   printf( (stat->st_mode & S_IXOTH) ? "x" : "-");
../Beispiel/find.c:97:2   printf("\t");
../Beispiel/find.c:98:2   printf("%d\t",(int) stat->st_size);
../Beispiel/find.c:104:2   printf("%s\t",timeBuffer);
../Beispiel/find.c:105:2   printf("%s\n",path);
../Beispiel/find.c:131:4     printf("could not open file %s", path);
../Beispiel/find.c:140:6       printf("%s:%d:%d %s", path, linenum, (strPosition - line), line);
../Beispiel/find.c:159:4     printf("Invalid parameter\n");
../Beispiel/find.c:160:4     printf("Usage: %s dir function\n", argv[0]);
../Beispiel/find.c:161:4     printf("\tfunction -print: shows the files and details\n");
../Beispiel/find.c:162:4     printf("\tfunction -grep content: shows the files with the given content\n");
../Beispiel/find.c:171:8         printf("Invalid parameter\n");
../Beispiel/find.c:172:8         printf("You have to give a search string\n");
../Beispiel/find.c:178:4     printf("invalid function\n");
../Beispiel/find.c~:25:4     printf("failed to open directory %s\n", dirname);
../Beispiel/find.c~:37:8         printf("out of memory\n");
../Beispiel/find.c~:42:7       sprintf(pathname,"%s/%s",dirname,entry->d_name);
../Beispiel/find.c~:55:8         printf("failed to stat %s\n", pathname);
../Beispiel/find.c~:88:2   printf( (stat->st_mode & S_IRUSR) ? "r" : "-");
../Beispiel/find.c~:89:2   printf( (stat->st_mode & S_IWUSR) ? "w" : "-");
../Beispiel/find.c~:90:2   printf( (stat->st_mode & S_IXUSR) ? "x" : "-");
../Beispiel/find.c~:91:2   printf( (stat->st_mode & S_IRGRP) ? "r" : "-");
../Beispiel/find.c~:92:2   printf( (stat->st_mode & S_IWGRP) ? "w" : "-");
../Beispiel/find.c~:93:2   printf( (stat->st_mode & S_IXGRP) ? "x" : "-");
../Beispiel/find.c~:94:2   printf( (stat->st_mode & S_IROTH) ? "r" : "-");
../Beispiel/find.c~:95:2   printf( (stat->st_mode & S_IWOTH) ? "w" : "-");
../Beispiel/find.c~:96:2   printf( (stat->st_mode & S_IXOTH) ? "x" : "-");
../Beispiel/find.c~:97:2   printf("\t");
../Beispiel/find.c~:98:2   printf("%d\t",(int) stat->st_size);
../Beispiel/find.c~:104:2   printf("%s\t",timeBuffer);
../Beispiel/find.c~:105:2   printf("%s\n",path);
../Beispiel/find.c~:131:4     printf("could not open file %s", path);
../Beispiel/find.c~:140:6       printf("%s:%d:%d\t%s", path, linenum, (strPosition - line), line);
../Beispiel/find.c~:159:4     printf("Invalid parameter\n");
../Beispiel/find.c~:160:4     printf("Usage: %s dir function\n", argv[0]);
../Beispiel/find.c~:161:4     printf("\tfunction -print: shows the files and details\n");
../Beispiel/find.c~:162:4     printf("\tfunction -grep content: shows the files with the given content\n");
../Beispiel/find.c~:171:8         printf("Invalid parameter\n");
../Beispiel/find.c~:172:8         printf("You have to give a search string\n");
../Beispiel/find.c~:178:4     printf("invalid function\n");
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$
```

1.3.7 Testfall 7 - Valgrind Funktion grep

Prüft, ob der allokierte Speicher wieder freigegeben wurde.

```
romanlum@ubuntu: ~/swo3/UebungMoodle4/Beispiel
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$ valgrind ./find .. -grep failed
==6047== Memcheck, a memory error detector
==6047== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==6047== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==6047== Command: ./find .. -grep failed
==6047==
../Beispiel/find.c:25:12            printf("failed to open directory %s\n", dirname);
../Beispiel/find.c:55:16                printf("failed to stat %s\n", pathname);
==6047==
==6047== HEAP SUMMARY:
==6047==     in use at exit: 0 bytes in 0 blocks
==6047==   total heap usage: 194 allocs, 194 frees, 408,561 bytes allocated
==6047==
==6047== All heap blocks were freed -- no leaks are possible
==6047==
==6047== For counts of detected and suppressed errors, rerun with: -v
==6047== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
romanlum@ubuntu:~/swo3/UebungMoodle4/Beispiel$
```