Dr. Johannes Mayer
Axel Blumenstock
Christian Ehrhardt

Allgemeine Informatik III / Systemnahe Software I (WS 2005/2006)

Abgabetermin: 7. Februar 2006

Wer suchet... (10 Punkte)

Unser Programm im letzten Blatt konnte bereits einen Verzeichnisbaum durchlaufen, um eine Häufigkeitsverteilung über die Anfangsziffern der Dateigrößen zu erstellen. Mit wenig Aufwand wollen wir dieses Programm nun zu einer vereinfachten Implementierung des Unix-Kommandos `find` ausbauen.
Unser `find` soll als Programmoptionen eine beliebige Zahl von Bedingungen (auch: Prädikaten), gefolgt von einer beliebigen Zahl von Verzeichnissen entgegennehmen und alle Dateien in diesen Verzeichnissen (und Unterverzeichnissen) ausgeben, die *allen* gegebenen Bedingungen genügen. Ohne Bedingung werden also alle Dateien ausgegeben; jedes weitere Prädikat schränkt die Ergebnismenge (möglicherweise) ein. Wird kein Verzeichnis angegeben, soll dafür das aktuelle Verzeichnis verwendet werden.
Das Programm soll folgende Bedingungen interpretieren können:

- `-n` *name* — findet Dateien mit dem gegebenen Namen
- `-m` *n*+ — findet Dateien, die *n* Tage oder älter sind (abzulesen an `st_mtime`)
- `-m` *n*− — findet Dateien, die *n* Tage oder jünger sind
- `-s` *n*+ — findet Dateien, die *n* oder mehr Bytes groß sind
- `-s` *n*− — findet Dateien, die *n* oder weniger Bytes groß sind
- `-p` *o* — findet Dateien, die mindestens die Zugriffsrechte *o* aufweisen. *o* ist die Oktaldarstellung des entsprechenden Bitmusters in `st_mode`.
- `-t` *typ* — findet Dateien, die den gegebenen Typ haben. Ein Typ `f` steht für reguläre Dateien, ein `d` für Verzeichnisse, ein `l` für symbolische Links, und so weiter.[1]

[1] Die weiteren Dateitypen finden Sie bei Bedarf in den libc-Infoseiten.

Optionen können sehr bequem mit Hilfe der Funktion `getopt()` verarbeitet werden. Gibt der Benutzer unbekannte Optionen oder ungültige Optionsargumente an, soll das Programm wie üblich mit einer Usage-Meldung reagieren.
Ein paar Beispielaufrufe:[2]

```
find
```

gibt alle Dateien in diesem und allen Unterverzeichnissen aus.

```
find -n README /usr /tmp
```

gibt alle Dateien namens „README“ in den Verzeichnissen `/usr` oder `/tmp` oder jeweils darunter aus.

```
find -nREADME -n LIESMICH
```

gibt nichts aus, da kein Verzeichniseintrag beide Bedingungen gleichzeitig erfüllen kann.

```
find -m 4+ -m 8- -m 12-
```

findet alle Dateien, die vor mindestens vier, aber höchstens acht Tagen verändert worden sind.

```
find -td -p002 /
```

findet global alle Verzeichnisse, für die jeder Schreibrechte hat, die also mindestens die Rechte `-------w-` haben (002 oktal ist 000 000 010 binär).

Fingerzeige zur Umsetzung[3]

Es ist möglicherweise am einfachsten, die Variablen, die die zu erfüllenden Bedingungen repräsentieren, mit „neutralen“ Werten zu initialisieren und bei der Optionsverarbeitung sukzessive zu setzen. Mehrfach angegebene Optionen oder gar Widersprüche können an dieser Stelle bereits erkannt werden. Fasst man alle diese Variablen in einer `struct` zusammen und gibt einen Zeiger darauf an die Funktion zum rekursiven Verzeichnisdurchlauf, lassen sich globale Variablen recht elegant vermeiden.
Oktalzahlen lassen sich problemlos mit `sscanf(s, "%o", &i)` interpretieren. Auf Zeichenketten wie „99+“ passt ein Formatstring wie `"%d%c"`. Beachten Sie, dass das Feld `st_mode` in `struct stat` sowohl den Typ, als auch die Zugriffsrechte bitcodiert enthält. Der Test, ob eine Datei mindestens die geforderten Zugriffsrechte aufweist, ist mit einer einzigen Bitoperation möglich.

Zusatzaufgabe (5 Punkte)

Nutzen Sie die Funktionen `regcomp()`, `regexec()` und `regfree()`, um mit einer Option `-N` auch eine Namenssuche mit regulären Ausdrücken zu anzubieten. Mehr Information zu diesen Funktionen finden Sie in den libc-Infoseiten.

[2] Wenn Ihre ausführbare Datei `find` heißt, geben Sie Acht, dass Sie auch `./find` und nicht `/usr/bin/find` aufrufen – darüber entscheidet im Zweifelsfalle Ihr $PATH.
[3] Hätte auch „Implementierungstips“ heißen können.

Nützliche Bibliotheksfunktionen

- `int getopt(int argc, char *argv[], char *optstring)` erhält als erste zwei Argumente genau die Argumente der `main()`-Funktion. Das dritte listet die (Ein-Zeichen-)Optionen, die erlaubt sein sollen. Nach jedem Optionszeichen, das zusätzlich einen Parameter erfordern soll, steht ein `':'`.

  Der Rückgabewert von `getopt()` ist:

  - das Zeichen der nächsten erkannten Option (gesteuert durch einen globalen Index namens `optind`);
  - `'?'`, falls eine unbekannte Option gelesen wurde;
  - `':'`, falls eine Option, die ein Argument erwartet, keines bekommen hat;
  - -1 am Ende der Optionsverarbeitung.

  Im Falle einer Option mit `':'` liegt das Optionsargument in der globalen Variablen `optarg` und muss von dort für die spätere Verwendung dupliziert werden – es sei denn, es wird direkt weiterverarbeitet, beispielsweise per `sscanf()` als Zahl interpretiert.

  In der Standardeinstellung kümmert sich `getopt()` auch um die Fehlermeldungen.

- `long time(time_t *)` liefert die Zahl der seit 1. Januar 1970 verstrichenen Sekunden. Der Parameter kann einfach NULL sein. Die Zeitfelder in `struct stat` werden ebenfalls in Sekunden seit 1970 angegeben.

Viel Erfolg!