Wichtig: Lesen Sie auch den Teil "**Hinweise zur Aufgabe**" auf diesem Blatt; Spezifikationen in diesem Teil sind ebenfalls einzuhalten!

Aufgabe 6: crawl (14.0 Punkte)

Implementieren Sie ein Programm `crawl`, das ähnlich dem UNIX-Kommando `find` arbeitet. Das Programm wird wie folgt aufgerufen:

```
crawl path... [-maxdepth=n] [-name=pattern] [-type=d,f] [-size=[+-]n]
```

Das Programm erhält einen oder mehrere Pfadnamen (Datei- oder Verzeichnisnamen) auf der Kommandozeile, optional gefolgt von Parametern und Suchausdrücken.

a) Makefile (2 Punkte)

Erstellen Sie ein zur Aufgabe passendes Makefile. Neben der Erzeugung der Objekt- und Binärdateien soll das Makefile die Pseudo-Targets `all` und `clean` unterstützen. Wird kein Argument an das **make(1)**-Programm übergeben, soll die ausführbare Datei `crawl` erzeugt werden.

b) `argumentParser`-Modul (4 Punkte)

Erstellen Sie ein Modul zum Verarbeiten der Kommandozeilen-Argumente, welches Optionen mit dem Format `-key=value` unterstützt. Die Schnittstelle ist in der Datei `argumentParser.h` vorgegeben und darf nicht verändert werden. Beachten Sie außerdem, dass die Funktionen des Moduls keine allgemeine Fehlerbehandlung zulassen – insbesondere dürfen also keine **malloc(3)**-Aufrufe oder vergleichbare Funktionen innerhalb des Modules verwendet werden.

c) `crawl`-Modul (8 Punkte)

Alle auf der Kommandozeile übergebenen Pfade werden rekursiv in einer Tiefensuche durchlaufen. Die speziellen Einträge `.` und `..` werden bei der Tiefensuche ignoriert, können aber als Pfade auf der Kommandozeile übergeben werden und dienen dann als Wurzelverzeichnis der Tiefensuche. Verzeichniseinträge, bei denen es sich nicht um ein Verzeichnis oder eine reguläre Datei handelt, werden sowohl bei der Tiefensuche als auch auf der Kommandozeile ignoriert. Alle nicht ignorierten Einträge werden mit ihrem relativen Pfad auf die Standardausgabe in jeweils einer eigenen Zeile ausgegeben.

Die `maxdepth`-Option erlaubt die Begrenzung der Suchtiefe auf ein frei wählbares Level $n \geq 0$ (0: Nur die auf der Kommandozeile übergebenen Pfade selbst werden untersucht, aber nicht deren Inhalt; 1: der Inhalt auf der Kommandozeile übergebener Verzeichnisse wird untersucht, aber nicht der ihrer Unterverzeichnisse, usw.). Ist diese Option nicht angegeben, so soll das Programm bis an die Blätter der Verzeichnishierarchie absteigen.
Ihr Programm soll außerdem eine Reihe ausgewählter Suchausdrücke unterstützen. Nur Pfadnamen, die alle angegebenen Bedingungen erfüllen, werden ausgegeben.
Achtung: Die Suchausdrücke schränken nur die Ausgabe von `crawl` ein. Wenn also die Suchausdrücke nicht auf ein gefundenes Verzeichnis passen, die maximale Pfadtiefe aber noch nicht erreicht ist, so wird dieses Verzeichnis dennoch durchsucht.

- `-name`: Mit diesem Ausdruck kann der Name eines Verzeichniseintrages (also nur die letzte Komponente des Pfades) mit einem Shell-Wildcard-Muster verglichen werden. Verwenden Sie für den Vergleich die Funktion **fnmatch(3)**.
- `-type`: Hiermit kann die Ausgabe auf Verzeichnisse (`d`) oder reguläre Dateien (`f`) eingegrenzt werden.
- `-size`: Hiermit kann nach Einträgen mit einer bestimmten Dateigröße (in Bytes) gesucht werden. Wird als Präfix das Zeichen - bzw. + vor der Größenangabe verwendet, so wird nach Einträgen gesucht, die kleiner bzw. größer als die angegebene Größe sind.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `Makefile, argumentParser.c, crawl.c`
- Hilfreiche *Manual-Pages*: **atoi(3)**, **basename(3)**, **closedir(3)**, **fnmatch(3)**, **lstat(2)**, **opendir(3)**, **readdir(3)**
- Zum Testen Ihres Programms eignet sich das Verzeichnis `/usr/share/doc`.
- Im Verzeichnis `/proj/i4sp1/pub/aufgabe6` finden Sie eine `crawl`-Implementierung, mit der Sie Ihre Lösung vergleichen können.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 11 Werktage (ohne Wochenenden und Feiertage)
Abgabezeit: 17:30 Uhr