Aufgabe 5: crawl (12.0 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.
Alle 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. Alle 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 vollen Pfad auf die Standardausgabe in jeweils einer eigenen Zeile ausgegeben.

a) Maximale Tiefe der Suche (`-maxdepth n`)

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 (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 Ihr Programm bis an die Blätter der Verzeichnishierarchie absteigen. Sie können davon ausgehen, dass die maximale Pfadtiefe die maximale erlaubte Zahl von offenen Dateien eines Prozesses nicht übersteigt (andernfalls soll sich das Programm mit einer Fehlermeldung beenden).

b) Suchausdrücke

Ihr Programm soll 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 gefundes 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: `crawl.c`
- Hilfreiche *Manual-Pages*: **atoi(3)**, **basename(3)**, **closedir(3) fnmatch(3)**, **lstat(2)**, **opendir(3)**, **readdir(3)**
- Beim Verarbeiten der Kommandozeilenargumente müssen diese nicht auf Validität überprüft werden. Dennoch muss Ihr Programm mit syntaktisch fehlerhaften Kommandozeilenargumenten umgehen können.
- Starten Sie `crawl` nicht auf den Sunray-Servern im CIP (*faui0sr0*, *faui0sr1* und Thin-Clients) und nicht auf den Rechnern *faui01**, *faui07** und *faui09**.
- Zum Testen Ihres Programms eignet sich das Verzeichnis `/usr/share/doc`.
- Im Verzeichnis `/proj/i4sp1/pub/aufgabe5` finden Sie eine `crawl`-Implementierung, mit der Sie Ihre Lösung vergleichen können.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 11 Werktage
Abgabezeit: 17:30 Uhr