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

Aufgabe 6: creeper (8.0 Punkte)

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

`creeper path... [-maxdepth=n] [-name=pattern] [-type=d,f]`

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

```
user@host:~$ ./creeper /usr/share/doc -maxdepth=2 -name=gcc
```

a) Makefile

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 `creeper` erzeugt werden. Greifen Sie dabei stets auf Zwischenprodukte (z. B. `creeper.o`) zurück. Das Makefile soll ohne eingebaute Regeln funktionieren (deshalb **make(1)** mit den Optionen `-rR` starten). Nutzen Sie die auf der Webseite genannten Compilerflags.

b) Die Hauptfunktion

Die Hauptfunktion (`main`) kümmert sich um das Initialisieren des Befehlszeilenparsers und verarbeitet die übergebenen Optionen. Verwenden Sie hierzu den **vorgegebenen Befehlszeilenparser** (siehe Hinweise). creeper soll die folgenden Optionen unterstützen:

- `-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(3p)**.
- `-type`: Hiermit kann die Ausgabe auf Verzeichnisse (`d`) oder reguläre Dateien (`f`) eingegrenzt werden.
- `-maxdepth`: 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.

c) Die `creeper`-Funktion

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, die die angegebenen Bedingungen erfüllen, werden mit ihrem relativen Pfad auf die Standardausgabe in jeweils einer eigenen Zeile ausgegeben.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `Makefile` (2 Punkte), `creeper.c` (6 Punkte)
- Hilfreiche *Manual-Pages*: **strtol(3p)**, **basename(3p)**, **fnmatch(3p)**, **lstat(3p)**, **opendir(3p)**, **readdir(3p)**, **closedir(3p)**
- In Ihrem Git-Repository befindet sich in der Datei `argumentParser.h` die Schnittstelle des zu verwendenden Befehlszeilenparsers.
- Im CIP finden Sie unter `/proj/i4sp1/pub/aufgabe6` die Implementierung des zu verwendenen Befehlszeilenparsers in den Dateien `argumentParser.{o,h}` sowie eine Referenzimplementierung der `creeper`.
- Eine Online-Beschreibung der *argumentParser*-Schnittstelle ist neben der Aufgabenstellung über die Webseite abrufbar.
- Zum Testen Ihres Programms eignet sich das Verzeichnis `/usr/share/doc`.

Hinweise zur Abgabe:

Bearbeitung: Dreiergruppen
Bearbeitungszeit: 7 Werktage (ohne Wochenenden und Feiertage)
Abgabezeit: 17:30 Uhr