Aufgabe 1

Es wird ein Programm, mit dem Namen myFilter.c, benötigt, dass eine beliebige Text/Datei öffnet, daraus liest. Zusätzlich sollen Parameter eingelesen werden die, je nachdem welcher Parameter eingelesen wurde, eine Operation ausführen und das Ergebnis in die Standart-Ausgabe schreibt.

Den Text/Datei kann mithilfe eines Parameters oder ohne eingelesen werden.

**(myfilter -i inputfile) oder (myfilter inputfile)**

Parameter werden mit einem vorangestellten **–** gekennzeichnet und das Programm sollte in der Lage sein weiterer Parameter einzulesen. Diese bestimmen welche Operation ausgeführt werden soll. Wenn keine Parameter übergeben worden sind, soll das Programm eine usage()-Meldung ausgeben.

**(./myfilter [-c/-w/-l] [-i] inputfile)**

Diese Parameter sollen als Kennzeichnung für folgende Operationen dienen:

**-c** soll alle Zeichen zählen und diese in die Standart-Ausgabe schreiben.

**-w** soll alle Wörter zählen und diese in die Standart-Ausgabe schreiben.

**-l** soll alle Zeilen zählen und diese in die Standart-Ausgabe schreiben.

Dabei soll das Programm so konzipiert werden, dass später leicht weitere Parameter hinzugefügt werden können. Außerdem falls kein Text/Datei angegeben wird, soll das Programm seinen Input aus der Standart-Eingabe beziehen und das Ergebnis in die Standart-Ausgabe schreiben.

Operationen der Anwendung

Das Programm soll mithilfe von Parametern verschiedene Operationen ausführen.

Dazu ist es zuerst einmal notwendig Parameter einzulesen. Bei diesem Programm habe ich mit der getopt()-Funktion, aus der unistad.h Libary, die Parameter als Fälle definiert und jedem Parameter einem Fall zugeteilt. Die Funktion bietet außerdem die Möglichkeit Fälle mit Argument zu übergeben. Hierbei war es sinnvoll die getopt()-Funktion mit einer while-Schleife auszustatten, sodass mehrere Parameter eingelesen werden. Dabei sei auch der Fall zu beachten, falls kein Parameter übergeben wird oder ein undefinierter Parameter übergeben wird soll eine Usage()-Meldung ausgegeben werden. Wenn ein Parameter eingelesen wird, soll dieser ein Signal/Flag setzen, um zu signalisieren das dieser Parameter vom Benutzer eingegeben wurde. Bei dem -i Parameter braucht man dieses nicht, da dieser Parameter nur ein Inputfile übergeben soll.

Parameter können nun eingelesen werden als nächstes habe ich mich an die Operation der einzelnen Parameter gemacht.

Der -c Parameter soll alle Zeichen einlesen und die Anzahl an Zeichen ausgeben. Testweise kann man sich eine Textdatei erstellen und mit der fgetc()-Funktion dieses in einem separaten Programm testen. Da die Aufgabe nicht genau wiedergab, ob Whitespace oder nur Buchstaben und Zahlen gezählt werden sollen habe ich mich entschieden jedes Zeichen zu zählen. Damit wir jedes Zeichen zählen können habe ich wieder eine while-Schleife mit der fgetc()-Funktion verbunden. Dabei muss ich darauf achten, dass solange der Text noch nicht zu Ende war (!feof()) die Schleife nicht aufhört zu zählen. Hierbei soll jedes Zeichen vom eingelesenen File eingelesen und zwischengespeichert werden. Nach jeder Änderung des Zwischenspeichers soll dann eine Zählvariable „Zeichen“ immer um 1 erhöht werden.

Der -w Parameter soll jedes Wort zählen und die Anzahl der Wörter ausgeben. Bei dem Wörter zählen soll das Programm erkennen, ob das zwischengespeicherte Zeichen ein Whitespace oder Buchstabe ist. Mit der isspace()-Funktion aus der ctype-h Libary werden Whitespace erkannt. Da nach einem Leerzeichen oder Zeilenumbruch normalerweise immer ein Wort reicht es ein Whitespace zu erkenne und dem Programm zu zeigen „hier war ein Whitespace jetzt befinden wir uns im Wort und erhöhe die Zählvariable „Word“ um 1.“ Nachdem ein Wort gezählt wurde, soll das Programm signalisiert bekommen „wir befinden uns nicht mehr im Wort“ obwohl wir nur das erste Zeichen erkannt haben und nicht pro Buchstaben die Word-Variable erhöht wird. Damit das letzte Wort der letzten Zeile als ein Wort gezählt wird muss das letzte Zeichen in der while-Schleife in einem anderen Zwischenspeicher gespeichert werden und überprüft werden, ob dieses Zeichen ein Buchstabe oder kein Buchstabe ist. Die isalpha()-Funktion auch aus der ctype.h Libary werden Buchstaben erkannt. Falls das letzte Zeichen als Buchstabe erkannt wird, soll ein Wort gezählt werden.

Der -l Parameter soll jede Zeile zählen und die Anzahl der Zeilen ausgeben. Hierbei habe ich die Zählvariable „Line“ jedes Mal um 1 zu erhöht, wenn ein Zeilenumbruch Zeichen vom Zwischenspeicher der while-Schleife erkannt wurde. Um jedoch die letzte Zeile als Zeile zu erkennen, da in der letzten Zeile eigentlich kein Zeilenumbruch mehr stattfindet hat es gereicht die Variable „Line“ um 1 zu erhöhen, wenn in der isalpha()-Funktion das letzte Wort erkannt wird.

Das Programm soll ein Inputfile mit oder ohne Parameter eingelesen werden.

Der -i Parameter soll nur die Textdatei übergeben. Da der Parameter nur ein Argument übergeben soll wird die „optarg“-Variable als Filename in einem Zeichenfeld temporär gespeichert werden.

Das Programm soll auch ohne den -i Parameter ein Inputfile übergeben bekommen. Falls ein Inputfile mit den Parametern -c,-w,-l übergeben wird muss dieses vom Programm eingelesen werden. Da diese Parameter auch ohne Argument auskommen würden muss das Programm die Anzahl der eingelesenen Argumente (argc) und die Anzahl der benötigten Argumente vergleichen. Wenn die eingelesenen Argumente mehr als die benötigten Argumente sind, soll der Index des nächsten Argumentes(argv[optind]) als Filename temporär gespeichert werden.

Bevor das Programm das Zählen anfängt, muss erst einmal überprüft werden, ob ein Inputfile übergeben worden ist und ob dieses auch lesbar und vorhanden ist. Dabei sei als erstes zu überprüfen, ob der Filename im Programm leer ist. Wenn dieser aber bereits einen Input erhalten hat, soll das File auf Lesbarkeit geprüft werden. Um die Lesbarkeit zu überprüfen habe ich einen Wert mit der open()-Funktion deklariert mit Abhängigkeit auf den Filenamen und die Lesbarkeit. Diesen Wert und die bereits vorhandene Structure aus der sys/types-h Library, die die Eigenschaften einer Datei überprüft, habe ich mit der fstat()-Funktion in eine Variable gespeichert, die mir bei Erfolg auf Lesbarkeit eine 0 als Wert oder eine 1 bei Fehlschlag wiedergibt. Falls der Wert nicht 0 sein sollte habe ich ein Error() ausgegeben und das Programm beenden. Bei Erfolg sollte die Datei geöffnet werden.