Stefan Kurtz **Universität Hamburg**
Abteilung für Genominformatik **Zentrum für Bioinformatik**

Programmierung für Naturwissenschaften 2
Sommersemester 2020
Übungen zur Vorlesung: Ausgabe am 26.05.2020

Aufgabe 5.1 (7 Punkte) Es gibt viele Anwendungen, die zeilenweise auf den Inhalt einer Datei zugreifen, z.B. wenn Zeilen sortiert oder wenn mehrfach zusammenfassende Operationen auf den Spalten eines Dokumentes angewendet werden müssen. In dieser Aufgabe sollen Methoden für den effizienten Zugriff auf die Zeilen einer Datei implementiert werden.

Beispiel: Die Datei, die im Editor wie folgt aussieht

```
MNIDDKL
SVLQ
```

liegt im Speicher als Folge von Zahlen (d.h. 8-bit ASCII codes) 77, 78, 73, 68, 68, 75, 76, 10, 83, 86, 76, 81, 10 vor, die den String `MNIDDKL\nSVLQ\n` repräsentieren. Das Ziel ist, die Zeilen in \0-terminierte Strings zu konvertieren (ohne sie zu kopieren) und ein Array mit Zeigern auf diese Strings zu erzeugen. In diesem Beispiel sind das zwei Zeiger: Der eine zeigt auf den String `MNIDDKL`, der andere auf den String `SVLQ`.

`PfNLineStore` ist der zentrale Typ in dieser Aufgabe. Die Funktionen, die auf diesem Typ basieren, werden in der Datei `pfn_line_store.c` implementiert. Es gibt keine anderen Funktionen, die die Interna der Struktur kennen. Daher sagt man auch, das `PfNLineStore` ein blickdichter (engl. opaque) Typ ist.

Die Datei `pfn_line_store.h` aus den Materialien enthält die Prototypen einiger zu implementierender Funktionen sowie die **`typedef`**-Deklaration für `PfNLineStore`. Wie üblich müssen Sie in der Datei `pfn_line_store.c` nach den notwendigen `include`-Anweisungen für die System-Headerdateien auch `pfn_line_store.h` inkludieren. Danach folgt diese **`struct`**-Definition

```
struct PfNLineStore
{
  size_t nextfree; /* number of lines */
  PfNLine *lines;  /* array with references to lines */
  char separator;
};
```

Dabei ist `PfNLine` ein Synonym für **`const char *`**.

Ihre Aufgabe ist es nun, die folgenden Funktionen zu implementieren:

- `PfNLineStore *pfn_line_store_new(unsigned char *file_contents, size_t size,char sep)`

  erzeugt aus dem Dateiinhalt mit `size` Bytes, referenziert durch `file_contents`, eine neue `PfNLineStore`-Struktur und liefert einen Zeiger auf die Struktur zurück.

  Die Struktur muss mit `malloc` dynamisch allokiert werden. Der Parameter `sep` gibt das Zeichen an, das nach jeder Einheit in der Datei folgt. In den meisten Anwendungen ist `sep`

das Zeichen \n, d.h. die Einheiten sind Zeilen. Zur Vereinfachung sprechen wir ab jetzt von Zeilen.

Am Ende der Funktion enthält `nextfree` in der genannten Struktur die Anzahl der Zeilen (2 im obigen Beispiel), und `lines` zeigt auf einen Speicherbereich mit `nextfree` vielen Einträgen des Basistyps `PfNLine`. Der i-te Eintrag zeigt auf den \0-terminierten String, der die i-te Zeile in der Datei repräsentiert (Nummerierung ab 0). Daher müssen im Speicherbereich, auf den `file_contents` zeigt, die Vorkommen von \n durch \0 ersetzt werden. Es darf keine Kopie des Dateiinhalts erzeugt werden.

Aus Effizienzgründen sollen alle Werte in einer einzigen Iteration über `file_contents` berechnet werden. Die effizienteste Möglichkeit besteht darin die Vorkommen von \n mit der Funktion `memchr`[1] zu bestimmen. In der selben Iteration müssen auch die Zeiger auf die Zeilenanfänge in `lines` gesetzt werden und die genannte Zeichenersetzung erfolgen.

Beachten Sie, dass die Anzahl der Einträge in `lines` nicht vorher bekannt ist. Sie müssen daher die Größe des Speicherbereichs, auf den `lines` zeigt, sukzessive mit `realloc` vergrößern, so wie es in `C_slides.pdf` im Abschnitt über `parsing and storing dates` an einem Beispiel gezeigt wurde.

- **`void`** `pfn_line_store_delete(PfNLineStore *pfn_line_store)` gibt den Speicherbereich für eine `PfNLineStore`-Struktur, referenziert über den Zeiger `pfn_line_store*`, wieder frei, falls `pfn_line_store` nicht `NULL` ist.
- `size_t pfn_line_store_number(`**`const`** `PfNLineStore *pfn_line_store)`, liefert die Anzahl der Zeilen, die durch die `PfNLineStore`-Struktur repräsentiert werden.
- `PfNLine pfn_line_store_access(`**`const`** `PfNLineStore *pfn_line_store,size_t i)` liefert die i-te Zeile aus der `PfNLineStore`-Struktur.
- **`char`** `pfn_line_store_sep(`**`const`** `PfNLineStore *pfn_line_store)` liefert das Separator-Zeichen, entsprechend der die Zerlegung des Dateiinhalts erfolgte, zurück.
- **`void`** `pfn_line_store_show(`**`const`** `PfNLineStore *pfn_line_store)` schreibt alle Zeilen, die durch die übergebene `PfNLineStore`-Struktur repräsentiert werden, mit einem abschließenden \n auf die Standard-Ausgabe.

In den letzten vier Funktionen wird jeweils auf Komponenten der Struktur zugegriffen, auf die `pfn_line_store` zeigt. Daher muss mit `assert` überprüft werden, dass `pfn_line_store` nicht `NULL` ist.

In den Materialien finden Sie das Hauptprogramm `pfn_line_store_mn.c` mit einem Optionsparser, der die C-Bibliotheksfunktion `getopt` nutzt. Schauen Sie sich die Funktionen und `pfn_line_store_options_new` und `pfn_line_store_options_delete` genauer an und beantworten Sie einige Fragen, die Sie im Programmcode finden.

In der Funktion `main` wird der Optionsparser aufgerufen, die durch den Benutzer spezifizierte Datei wird eingelesen und die entsprechende `PfNLineStore`-Struktur aufgebaut. Schließlich wird je nach Option für die Datei ein Wert bzw. der Dateiinhalt in veränderter Form ausgegeben. Am Ende erfolgt die Freigabe des Speichers.

Durch `make` kompilieren Sie Ihr Programm. Durch `make test` verifizieren Sie die Korrektheit für einige Testdateien.

[1] siehe `http://man7.org/linux/man-pages/man3/rawmemchr.3.html`

Punkte-Verteilung:

- für die Implementierung von `pfn_line_store_new`: 3.5 Punkte
- für die Implementierung aller anderen Funktionen: 1.5 Punkt insgesamt.
- für die akzeptable Beantwortung der Fragen: 2 Punkte

Zur Bearbeitung dieser Aufgabe sollten Sie die Abschnitte 1-6, 8, 9, 11, 14, 16, 19 (entsprechend der Tabelle in `pfn2_vorlesung_2020.html`) kennen.

Aufgabe 5.2 (3 Punkte)

In dieser Aufgabe geht es wieder um die Türme von Hanoi. Diesmal soll die Anzahl der benötigen Schritte, um n Scheiben zu bewegen, in Abhängigkeit von n berechnet werden. Um diese Zahl experimentell zu ermitteln, sollen Sie das C-Programm `hanoi.c` aus der Musterlösung in `hanoi_count.c` umbenennen und es so modifizieren, dass die Anzahl der Schritte in Abhängigkeit eines Eingabeparameters n ausgegeben wird. Benennen Sie die Funktion `hanoi_rec_print` um in `hanoi_rec_count`. Diese soll nun selbst keine Ausgabe mehr liefern. Stattdessen soll mit einer **`return`**-Anweisung die Anzahl der Scheibenbewegungen zurückgeliefert werden. Sie dürfen in `hanoi_rec_count` keine globalen Variablen verwenden und keine weiteren Parameter hinzufügen. Die Berechnung muss also alleine mit dieser Funktion und mit Hilfe Ihrer Rückgabewerte erfolgen.

Das Programm selbst bekommt als Parameter nun den maximalen Wert $nmax$ von n, berechnet die Anzahl der Schritte für alle n, $1 \leq n \leq nmax$ und gibt diese in tab-separierter Form auf stdout aus.

Durch `make` kompilieren Sie Ihr Programm. Durch `make test` verifizieren Sie die Korrektheit für $nmax = 20$.

Entwickeln Sie auf Basis der empirisch ermittelten Werte eine Gleichung, die die Anzahl der Schritte in Abhängigkeit von n angibt. Beweisen Sie die Korrektheit der Gleichung durch vollständige Induktion über n. Zur Erinnerung: Für eine vollständige Induktion müssen Sie zunächst die Korrektheit Ihrer Gleichung für den Basisfall zeigen, also für $n = 0$. Dann machen Sie eine Induktionsannahme, die darin besteht, dass Ihre Gleichung für $n > 0$ wahr ist. Sie zeigen nun unter dieser Annahme, dass die Gleichung auch für $n + 1$ gilt. Dann gilt sie nach dem Induktionsprinzip für alle n.

Punkteverteilung:

- Implementierung der rekursiven Funktion: 1 Punkt
- Bestimmung der Gleichung: 0.5 Punkte
- Induktionsbeweis: 1.5 Punkte

Zur Bearbeitung dieser Aufgabe sollten Sie die Abschnitte 1-6, 14, 17 (entsprechend der Tabelle in `pfn2_vorlesung_2020.html`) kennen.

Bitte die Lösungen zu diesen Aufgaben bis zum 09.06.2020 um 18:00 Uhr an pfn2@zbh.uni-hamburg.de schicken.