Übungen Betriebssysteme (BS)

A5 – Dateioperationen

https://moodle.tu-dortmund.de/course/view.php?id=34604

Peter Ulbrich

peter.ulbrich@tu-dortmund.de

https://sys.cs.tu-dortmund.de/EN/People/ulbrich/

technische universität
dortmund

Theoriefrage 1

- Welche Bedeutung haben die Verzeichnisse "*.*" und "*..*" in einem UNIX-Dateisystem?

Theoriefrage 1

- Welche Bedeutung haben die Verzeichnisse "*.*" und "*..*" in einem UNIX-Dateisystem?
- Jedes Verzeichnis beinhaltet zwei *spezielle* Verzeichnisse
  - „*.*“ (*dot*): Bezeichnet das aktuelle Verzeichnis
  - „*..*“ (*dotdot*): Bezeichnet das übergeordnete Verzeichnis

```
total 48K
drwxr-xr-x 2 al staff 4,0K Aug  2  2021 .
drwxr-xr-x 3 al staff 4,0K Jul  1 09:35 ..
-rw-r--r-- 1 al staff  12K Aug  2  2021 hallo.mp3
-rw-r--r-- 1 al staff  20K Aug  2  2021 sinus_ohne_id3.mp3
-rw-r--r-- 1 al staff 4,3K Aug  2  2021 telefonstreich.mp3
```

Theoriefrage 2

- Beschreibt mit eigenen Worten den Unterschied zwischen block- und zeichenorientierten Geräten.

Theoriefrage 2

- Beschreibt mit eigenen Worten den Unterschied zwischen block- und zeichenorientierten Geräten.
- Blockorientiert
  - Wahlfreier Zugriff auf Daten
  - Bsp.: Festplatten, USB-Sticks, DVDs
- Zeichenorientiert
  - Sequentieller, zeichenweiser Zugriff auf Daten
  - Bsp.: Tastatur oder Maus

a) Ausgeben von ID3v1 Tags

- Pfad zur Datei wurde an `idTagFile()` übergeben.
- Können wir sofort loslegen und die Tags auslesen?

Quelle: wikimedia.org (CC BY 4.0)

a) Ausgeben von ID3v1 Tags

- **Schritt 1**: Überprüfen, ob übergebener Pfad eine reguläre Datei betrifft
- Schritt 2: Überprüfen, ob diese Datei eine MP3-Datei ist
- Schritt 3: Überprüfen, ob Datei MP3-Tags enthalten kann (>128 Bytes)

```
void idTagFile(const char *fileName,char *comment)
{
    ...
    struct stat fileInfo = {0};

    if (stat(fileName,&fileInfo) == -1)
    {
        perror("stat");
        return;
    }
    ...
```

a) Ausgeben von ID3v1 Tags

- Schritt 1: Überprüfen, ob übergebener Pfad eine reguläre Datei betrifft
- **Schritt 2**: Überprüfen, ob diese Datei eine MP3-Datei ist
- **Schritt 3**: Überprüfen, ob Datei MP3-Tags enthalten kann (>128 Bytes)

```
…
   /* Es liegt eine regulaere Datei vor */
   if (S_ISREG(fileInfo.st_mode))
   {
      if (strncmp(fileName + strlen(fileName) -4,".mp3",4))
      {
         /* ... Fehlerbehandlung ... */
      }

      /* Die Datei ist zu klein um ID3-Informationen zu enthalten */
      if (fileInfo.st_size < 128)
      {
         /* ... Fehlerbehandlung ... */
      }
```

a) Ausgeben von ID3v1 Tags

- Passende MP3-Datei gefunden, was nun?
- Wie kommen wir an die Informationen?

a) Ausgeben von ID3v1 Tags

- **Schritt 1**: Öffnen der Datei
- **Schritt 2**: Setzen des Lesezeigers
- Schritt 3: Auslesen der letzten 128 Byte
- Schritt 4: Schließen der Datei

```
...
char buffer[ID3_SIZE];
...

filePointer = fopen(fileName,"r");
if (filePointer == NULL)
{
   /* ... Fehlerbehandlung ... */
}

/* Positionierung des Lesezeigers. */
if (fseek(filePointer,-128L,SEEK_END) == -1)
{
   /* ... Fehlerbehandlung ... */
}
```

a) Ausgeben von ID3v1 Tags

- Schritt 1: Öffnen der Datei
- Schritt 2: Setzen des Lesezeigers
- **Schritt 3**: Auslesen der letzten 128 Byte
- **Schritt 4**: Schließen der Datei

```
if (fread(buffer,sizeof(char),ID3_SIZE,filePointer) != ID3_SIZE)
{
   /* ... Fehlerbehandlung ... */
}
if (fclose(filePointer) == EOF)
{
   /* ... Fehlerbehandlung ... */
}
```

a) Ausgeben von ID3v1 Tags

- Ausgangspunkt: Die letzten 128 Bytes der MP3-Datei befinden sich in einem Puffer
- Können wir jetzt endlich die Tags auslesen?

Quelle: wikimedia.org (CC BY 4.0)

a) Ausgeben von ID3v1 Tags

- Ausgangspunkt: Die letzten 128 Bytes der MP3-Datei befinden sich in einem Puffer
- Können wir jetzt endlich die Tags auslesen?
- Nein!
- Vielleicht sind gar keine Tags vorhanden!

```
if (strncmp("TAG",buffer,3)) {
    /* Fehlerbehandlung */
}
```

a) Ausgeben von ID3v1 Tags

- Jetzt sind wir endlich fertig. Auslesen und ausgeben. Los geht's!

```
struct mp3file* mp3 = malloc( sizeof( struct mp3file ) );

/* Informationen übertragen */
strncpy( mp3->titel,     &buffer[  3], 30 );
strncpy( mp3->interpret, &buffer[ 33], 30 );
strncpy( mp3->album,     &buffer[ 63], 30 );
strncpy( mp3->jahr,      &buffer[ 93],  4 );
strncpy( mp3->kommentar, &buffer[ 97], 30 );

/* Strings terminieren */
mp3->titel[30] = '\0';
mp3->interpret[30] = '\0';
mp3->album[30] = '\0';
mp3->jahr[4] = '\0';
mp3->kommentar[30] = '\0';

mp3->genre = buffer[127];
```

- Am Ende den Speicher hinter mp3 wieder freigeben

b) Suchen in Verzeichnissen

- Aufgabe: Ausgabe aller ID3-Tags aller Dateien in einem Verzeichnis

Quelle: wikimedia.org (CC BY 4.0)

b) Suchen in Verzeichnissen

- Aufgabe: Ausgabe aller ID3-Tags aller Dateien in einem Verzeichnis

- Oh, ein Verzeichnis, was nun?

```
void idTagFile(const char *fileName,char *comment)
{
   ...
   struct stat fileInfo = {0};

   if (stat(fileName,&fileInfo) == -1)
   {
      perror("stat");
      return;
   }
   ... /* siehe a) */
   /* Es liegt eine regulaere Datei vor */
   else if (S_ISDIR(fileInfo.st_mode))
   {
      /* Was nun? */
   }
```

Quelle: wikimedia.org (CC BY 4.0)

b) Suchen in Verzeichnissen

- Schritt 1: Verzeichnis öffnen
- Schritt 2: Elemente des Verzeichnis durchlaufen
- Schritt 3: Gefundene Elemente behandeln
- Schritt 4: Verzeichnis schließen

```
void idTagDir(const char *dirName, char *comment) {
   struct dirent *entry = NULL;

   DIR *dir = opendir(dirName);
   if (dir == NULL)
   { /* ... Fehlerbehandlung ... */  }

   while ((entry = readdir(dir)))
   {
      /*Hier geht's weiter */
   }

   if (closedir(dir) == -1)
   { /* ... Fehlerbehandlung ... */  }
```

b) Suchen in Verzeichnissen

- Schritt 1: „.“ und „..“ filtern
- Schritt 2: Pfad für das gefundene Element zusammenbauen
- Schritt 3: Rekursiver Aufruf

```
while ((entry = readdir(dir)))
{
   if (!strncmp(".", entry->d_name, 2)
      || !strncmp("..", entry->d_name, 3))
                  continue;

   fullpath = malloc(strlen(dirName) + strlen(entry->d_name) + 2);
   if (fullpath == NULL){ /* ... Fehlerbehandlung ... */ }

   strcpy(fullpath,dirName);
   strcat(fullpath,"/");
   strcat(fullpath,entry->d_name);

   idTagFile(fullpath,comment);

   free(fullpath);
}
```

b) Suchen in Verzeichnissen

- Schritt 1: „.“ und „..“ filtern
- **Schritt 2**: Pfad für das gefundene Element zusammenbauen
- Schritt 3: Rekursiver Aufruf

```
while ((entry = readdir(dir)))
{
   if (!strncmp(".", entry->d_name,
      || !strncmp("..", entry->d_name, 3))
                  continue;

   fullpath = malloc(strlen(dirName) + strlen(entry->d_name) + 2);
   if (fullpath == NULL){ /* ... Fehlerbehandlung ... */ }

   strcpy(fullpath,dirName);
   strcat(fullpath,"/");
   strcat(fullpath,entry->d_name);

   idTagFile(fullpath,comment);

   free(fullpath);
}
```

Genügend Speicherplatz allozieren

Pfad zusammen bauen

Speicherplatz freigeben

b) Suchen in Verzeichnissen

- Schritt 1: „.“ und „..“ filtern
- Schritt 2: Pfad für das gefundene Element zusammenbauen
- **Schritt 3**: **Rekursiver** Aufruf

```
while ((entry = readdir
{
   if (!strncmp(".", en
      || !strncmp("..",
                    cont

   fullpath = malloc(str                                  ) + 2);
   if (fullpath == NULL){ /*          rbehandlung ... */ }

   strcpy(fullpath,dirName);
   strcat(fullpath,"/");
   strcat(fullpath,entry->d_   me);

   idTagFile(fullpath,comment);

   free(fullpath);
}
```

Erlaubt das **rekursive** Absteigen in Verzeichnisse.
(War nicht gefordert; Vorgabe war aber so ausgerichtet)

c) Bearbeiten von Tags

- Nur lesen ist langweilig. Keine Interaktion. Ich will Tags schreiben!
- Schritt 1: Neues Argument für den Aufruf definieren

```
    if (argc >= 3)
    {
       if (!strncmp("--modify",argv[1],9))
       {
          do
          {
             printf("Bitte gib' einen Kommentar ein(max. 30 Zeichen):");
             retval = scanf("%30s",buffer);
             while (getchar() != '\n');
          }
          while (!retval);
          temp = buffer;
          i = 2;
       }
    }
…
    idTagFile(argv[i],temp);
```

c) Bearbeiten von Tags

- Schritt 1: Datei öffnen (schon erledigt)
  - Kurze Zwischenfrage: Welcher Modus?
- Schritt 2: Schreibzeiger positionieren
- Schritt 3: Kommentar schreiben
- Schritt 4: Datei schließen

```
void idTagFile(const char *fileName,char *comment)
{
...
        if (fseek(filePointer,-31L,SEEK_END) == -1)
        { /* ... Fehlerbehandlung ... */ }

        if (fwrite(comment, sizeof(char),
            (strlen(comment) < 30) ?
             strlen(comment) + 1 : 30, filePointer) == -1)
        { /* ... Fehlerbehandlung ... */ }
...
}
```