

Aufgabe 5: piper (12.0 Punkte)

Schreiben Sie ein mehrfädiges Programm **piper** (**pipe** multiplier), welches zeilenweise vom Standardeingabekanal `stdin` liest und die gelesenen Zeilen an eine Menge von UNIX-Pipes weiterleitet. Das Programm wird wie folgt aufgerufen:

```
piper <pipes...>
```

Bei Zugriffen auf globale Datenstrukturen muss auf ausreichende Synchronisierung geachtet werden. Langsame, potentiell blockierende Operationen (z. B. **printf(3)**, **fgets(3)**) dürfen dabei nicht in kritischen Abschnitten ausgeführt werden.

Haupt-Thread

Jede der als Parameter übergebenen UNIX-Pipes (*pipes*) wird jeweils an einen neu zu erzeugenden Thread (genannt *Schreib-Thread*) übergeben. Nach der Initialisierungs-Phase liest der Haupt-Thread zeilenweise (**fgets(3)**) maximal 100 Zeichen von `stdin` ein und signalisiert allen aktiven (s. u.) Schreib-Threads das Vorhandensein neuer Daten. Anschließend wartet er passiv, bis alle aktiven Schreib-Threads die Zeile geschrieben haben, bevor er die nächste Zeile einliest.

Wird das Ende der Eingabe erreicht, so beendet der Haupt-Thread den gesamten Prozess.

Schreib-Thread

Jeder Schreib-Thread stellt zunächst sicher, dass es sich bei dem als Argument übergebenen Dateinamen um eine Pipe handelt, und öffnet diese zum Schreiben. Sollte es keine Pipe sein, so beendet sich der Thread. Das Öffnen einer UNIX-Pipe blockiert solange, bis diese (von einem anderen Prozess) zum Lesen geöffnet wurde. Nach dem erfolgreichen Öffnen seiner Pipe registriert sich jeder Schreib-Thread mit Hilfe der Funktion `sbufAddSem()` aus dem Modul `sbuf` beim Haupt-Thread und wird damit „aktiv“. Anschließend gibt er eine Meldung, die den Dateinamen und die ID aus dem `sBuf`-Modul enthält, auf `stderr` aus und wartet passiv auf neue Daten vom Haupt-Thread. Sind Daten vorhanden, werden diese in die Pipe geschrieben und die Ausgabe mit **fflush(3)** erzwungen. Anschließend wird der Haupt-Thread über das erfolgreiche Schreiben benachrichtigt.

sbuf-Modul

Das `sbuf`-Modul verwaltet ein Array, in das Semaphore eingefügt werden können. Jeder eingefügte Semaphor erhält eine eindeutige ID, die dem Index seines Platzes im Array entspricht. Mit Hilfe der Funktion `sbufGetSem()` kann man auf einen beliebigen Semaphor zugreifen.

Das Entfernen von Semaphoren aus dem Puffer ist nicht vorgesehen.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `piper.c`, `sbuf.c`
- Zum Übersetzen des Programmes ist das zusätzliche Compiler- und Linker-Flag `-pthread` notwendig.
- Verwenden Sie für die Synchronisation das vorgegebene Semaphor-Modul (`sem.o`, `sem.h`).
- UNIX-Pipes können mit `mkfifo <name>` angelegt werden. Siehe **mkfifo(1)**.
- Zum Testen Ihrer Implementierung können Sie die erzeugten Pipes mit Hilfe des Kommandos **cat(1)** öffnen und deren Inhalt ausgeben lassen.
- Das Schreiben auf eine UNIX-Pipe, die nicht *mehr* zum Lesen geöffnet ist, da sich z. B. der lesende Prozess beendet hat, führt zum Abbruch des Programmes. Für die `piper` ist dieses Verhalten gewünscht. Wie dieses Verhalten beeinflusst werden kann, wird in Systemprogrammierung 2 behandelt.
- Bei den auszugebenden Meldungen können Sie sich an der Referenzimplementierung in `/proj/i4sp1/pub/aufgabe5` orientieren.

Hinweise zur Abgabe:

Bearbeitung: Einzel

Bearbeitungszeit: 14 Werktage (ohne Wochenenden und Feiertage)

Abgabezeit: 17:30 Uhr