Wilhelm-Schickard-Institut für Informatik

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Jens Haußmann, M. Sc.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

05.03.2018

Wintersemester 2017/2018
Praktikum Betriebssysteme - Tag 5

IPC – Mutex

Die für heute benötigte Datei `tsl.o` finden Sie als Download im Ilias.

Aufgabe 5.1 *(Mutex mit TSL aus Library implementieren)*

Implementieren Sie in einer Datei `mutex.c` die Funktionen `mutex_lock()` und `mutex_unlock()`. Benutzen Sie dafür die gegebene „Test and Set Lock“ Instruktion `tsl`:

```
int tsl(unsigned char *adr);
```

Deren Implementierung befindet sich in `tsl.o`. *(Hinweis: Diese Object-Datei ist für die AMD64-Architektur implementiert; die Pool-Rechner (CS-Rechner) sind mit den entsprechenden Prozessoren ausgestattet.)* Erstellen Sie außerdem eine dazugehörige Header-Datei `mutex.h`.

Aufgabe 5.2 *(Mutex-Implementierung testen)*

Schreiben Sie ein Testprogramm `test_mutex.c`, bei dem zwei Threads gleichzeitig versuchen auf einen kritischen Bereich zuzugreifen. Verwenden Sie Ihre Implementierung `mutex_lock()` und `mutex_unlock()` um den Zugriff zu regeln.

Aufgabe 5.3 *(TSL selbst implementieren)*

Als nächstes sollen Sie die gegebene TSL-Implementierung durch eine eigene ersetzen (Assembler-Code).

a) Übersetzen Sie zunächst folgendes Programm `test.c` mit den Compileroptionen `-O2 -S -o test.s`. Finden Sie anhand der Ausgabe in `test.s` heraus, wie die Parameter- und Funktionswertübergabe in C-Programmen auf Assemblerebene realisiert sind.

```
/* test.c */
int test(unsigned char *adr)
{
  if (*adr) {
    (*adr)++;
  }
  return 0;
}
```

b) Das atomare „Test and Set Lock“ lässt sich auf AMD64-Prozessoren mit der Anweisung `lock xchg` realisieren. Implementieren Sie – ausgehend von `test.s` – die Funktion `tsl()` in Assembler. Speichern Sie das Ergebnis in einer Datei `my_tsl.s` und übersetzen Sie Ihre Routine mit dem Programm `as` in eine Datei `my_tsl.o`.

Aufgabe 5.4 *(Mutex mit `my_tsl` implementieren)*

Implementieren Sie nun – analog zu Aufgabe 6.1 – in einer Datei `my_mutex.c` die Funktionen `mutex_lock` und `mutex_unlock` erneut und verwenden Sie dazu ihre eigene TSL-Implementierung aus Aufgabe 6.3b. Erstellen Sie außerdem wieder ein dazugehöriges Header-Datei `my_mutex.h`.

Aufgabe 5.5 *(Eigene Mutex-Implementierung testen)*

Schreiben Sie – analog zu Aufgabe 6.2 – ein Testprogramm `test_my_mutex.c` um die Mutex-Implementierung aus Aufgabe 6.4 zu testen.

Geben Sie bitte genau ein Verzeichnis ab, das genau die folgenden Dateien enthält:

- `mutex.c` (6.1)
- `mutex.h` (6.1)
- `tsl.o` (6.1)
- `test_mutex.c` (6.2)
- `test.c` (6.3a)
- `my_tsl.s` (6.3b)
- `my_mutex.c` (6.4)
- `my_mutex.h` (6.4)
- `test_my_mutex.h` (6.5)
- `Makefile`

Legen Sie das Makefile so an, dass durch das Shell-Kommando `make` alle für dieses Aufgabenblatt zu erzeugenden Dateien erstellt werden. Unter anderem sind das die Dateien:

- `mutex.o` (6.1)
- `test_mutex` (6.2)
- `test.s` (6.3a)
- `my_tsl.o` (6.3b)
- `my_mutex.o` (6.4)
- `test_my_mutex` (6.5)

Achten Sie darauf, dass `make` die Dateien im selben Verzeichnis erzeugt, in dem sich auch die Quelldateien befinden. Durch das Shell-Kommando `make clean` soll es möglich sein alle durch `make` erstellten Dateien wieder zu löschen.