Institut für Technische Informatik
und Kommunikationsnetze

Prof. Dr. B. Plattner, ETH Zentrum, ETZ G-89, Gloriastrasse 35, Zürich, 30.3.01
CH-8092 Zürich, e-mail: plattner@tik.ee.ethz.ch

Technische Informatik 2, SS01

TOPSY Kurzreferenz

WWW: `http://www.tik.ee.ethz.ch/tik/education/lectures/TI2/topsyref.html`
TOPSY Website: `http://www.tik.ee.ethz.ch/~topsy/`

1 Installieren von TOPSY

Gehen Sie in Ihr Heimverzeichnis (home directory):

```
cd
```

Für den Compiler müssen die richtigen Pfade gesetzt werden. Dies geschieht durch ein Startup-Skript, welches immer beim Einloggen automatisch ausgeführt wird. Das Startup-Skript muss also nur einmal mit diesen Informationen ergänzt werden. Falls Sie Ihren Account bereits im letzten Semester für die Praktika Technische Informatik 1 benutzt haben, haben Sie das damals schon gemacht und müssen es hier nicht wiederholen. Ansonsten führen Sie folgenden Befehl einmal aus:

```
echo source /home/tinf/TechnischeInformatik1/Installation/.cshrc >> ~/.cshrc
```

Kopieren Sie nun Topsy in Ihr Heimverzeichnis (sämtliche Topsy-Files sind in einem einzigen File Topsy.tar eingepackt):

```
cp /home/tinf/TechnischeInformatik2/Topsy.tar ~
```

Starten Sie das Programm *tar* (create tape archives and add or extract files) wie folgt, um Topsy auszupacken:

```
tar -xvf Topsy.tar
```

Sie erhalten nun ein Verzeichnis “Topsy”, das in mehreren Unterverzeichnissen den gesamten *Sourcecode* des Topsy-Systems enthält. Für jedes Modul, das in Kapitel 2.2 der Topsy Dokumentation aufgeführt ist, gibt es ein Unterverzeichnis mit den hardwareunabhängigen C-Files (Memory, Threads, IO, Topsy (mit Syscalls), Startup, User). In jedem Unterverzeichnis stehen in einem weiteren Unterverzeichnis “mips” die hardwareabhängigen Teile. Im Unterverzeichnis “User” werden Sie Ihre Topsy-Programme ablegen; einige Beispielprogramme sind dort bereits vorhanden, ebenso wie die Topsy-Shell und das Modul “UserSupport” mit Hilfsfunktionen.

Gehen Sie in das Verzeichnis “Topsy”:

```
cd Topsy
```

Dort wird mit dem Programm *gmake*, einer Variante von *make* (maintain, update, and regenerate related programs and files), das Betriebssystem kompiliert und gelinkt. Dies erfolgt mit dem Befehl:

```
gmake
```

2 Verwenden eigener Programme

Hier wird Schritt für Schritt beschrieben, was zu tun ist, um ein Programm zu kompilieren und zu linken (wie in der Topsy Dokumentation Kapitel 7.1.1 genauer beschrieben). Ihr Programm sollte sich im Verzeichnis “User” befinden; für “Hello World” sind alle Schritte als Beispiel bereits durchgeführt worden.

Im Verzeichnis “Topsy”, gehen Sie in das Verzeichnis “User”:

```
cd User
```

Die zwei Files “HelloWorld.c” und “HelloWorld.h” sind schon vorhanden. Das C-File “HelloWorld.c” muss bestimmte Include-Files haben, ferner muss eine Startfunktion (z.B. Hello()) existieren, die beim Starten des Programms aufgerufen wird:

```
#include "../Topsy/Syscall.h"
#include "UserSupport.h"

void Hello() {
  .....
}
```

Im H-File “HelloWorld.h” steht die Deklaration dieser Startfunktion:

```
void Hello();
```

Nun muss diese Startfunktion im File “Shell.c” eingetragen werden, damit sie von der Shell aus gestartet werden kann, d.h. Ihr Programm wird so direkt in Topsy eingebaut. Es müssen die folgenden drei Parameter eingegeben werden: Der erste bezeichnet den Programmnamen, so wie er dann aus der Shell gestartet werden können soll (als Argument des Topsy-Befehls start). Der zweite Parameter bezeichnet den Namen der Startfunktion, wie sie in den C- und H-Files beschrieben ist. Mit dem letzten Parameter wird ein Array von Strings als Parameter zur Startfunktion des Threads übergeben. Die Konstante NUSERCOMMANDS muss auf die Anzahl eingetragener Funktionen gesetzt werden. Damit der Startfunktionsname in diesem File bekannt ist, muss am Anfang das H-File included werden:

```
#include "HelloWorld.h"
.....
#define NUSERCOMMANDS        8
.....
ShellFunction userCommands[NUSERCOMMANDS] = {
  {"shell", main, (ThreadArg)argArray},
  {"hello", Hello, (ThreadArg)argArray}
  .....
}
```

Damit Ihr Programm auch kompiliert und gelinkt wird, muss es im File “Makefile” im Verzeichnis “User” eingetragen werden. Dies ist notwendig, damit gmake darüber informiert wird, dass “HelloWorld.c” nun Teil der zu übersetzenden Software ist:

```
USERFILES= HelloWorld.c .....
```

Nun kann das System mit gmake im “Topsy” Verzeichnis neu kompiliert und gelinkt werden. Danach sind auch Ihre Programme im System enthalten:

```
cd ..
gmake
```

3 Programme starten

Topsy läuft auf dem MIPS-Praktikumsboard, das Sie schon von den Praktika Technische Informatik 1 her kennen, und wird wiederum mit dem Debugger mips-gdb auf das Board geladen und ausgeführt. Das MIPS-Praktikumsboard ist über zwei Kanäle mit Ihrer tardis Sun-Workstation verbunden: der Debugger kommuniziert über “ttyb”, um Topsy zu laden und zu starten, während Topsy selbst, sobald es läuft, über “hardwire-a” kommuniziert.

Öffnen Sie daher eine zweite UNIX-Shell (xterm oder cmdtool) mit der Maus oder mit dem Befehl:

```
xterm &
```

Dort starten Sie das Programm tip, welches eine Terminal-Verbindung mit dem MIPS-Board startet (dieses vorher einschalten), über die Topsy kommunizieren wird, wie folgt:

```
tip hardwire-a
```

In der ersten UNIX-Shell können Sie nun (im Verzeichnis “Topsy”) den Debugger starten mit dem Topsy-Kernel als Argument:

```
mips-gdb topsy.ecoff
```

Dann geben Sie dort die folgenden Gdb-Kommandos ein, um Topsy zu laden und zu starten:

```
target mips /dev/ttyb
load
run
```

Nun sollte in der zweiten UNIX-Shell, wo tip läuft, die Topsy-Shell erscheinen (die Meldung “Programming FPGA ... Error occurred!!!” nicht beachten). Dort können Sie Befehle der Topsy-Shell eingeben (siehe Topsy Dokumentation, Kapitel 7.1.1), um die im Topsy vorhandenen oder von Ihnen eingebauten Programme zu starten, z.B. das “Hello World” Programm:

```
start hello
```

oder das “ThreadsDemo” Programm:

```
start demo
```

Zum Beenden von tip geben Sie ein:

```
~.
```

Zum Beenden von mips-gdb drücken Sie Ctrl-Z und geben anschliessend `kill %1` ein (bzw. verwenden kill % mit der aktuellen job-Nummer von mips-gdb, wie durch den Befehl `jobs` angezeigt).

Zum Reset das MIPS-Praktikumsboard ausschalten.

4 Topsy-Shell-Befehle

Siehe auch Topsy Dokumentation, Kapitel 7.1.

`start <function>`	Programm starten, z.B. `start hello` oder `start shell`
`start <function> &`	Programm im Hintergrund starten, z.B. `start demo &`
`ps`	Liste der Threads anzeigen
`kill <threadid>`	Thread beenden (Threadid wie mit `ps` angezeigt verwenden), z.B. `kill 8`
`exit`	Shell verlassen
`help`	Liste der Befehle anzeigen

5 Übersicht der Hilfsfunktionen: UserSupport.h

```
/* copy nbBytes from address srcAddress to address targetAddress
*/
void byteCopy( Address targetAddress, Address sourceAddress,
  unsigned long int nbBytes);

/* fill memory with zeros */
void zeroOut(Address target, unsigned long int size);

/* copy a NULL-terminated string from source to target */
void stringCopy( char* target, char* source );

/* copy a string or at most the first size-1 bytes */
/* and terminate it with 0 */
void stringNCopy( char* target, char* source,
  unsigned long int size);

/* string length */
int stringLength( char* s);

/* test and set memory in one atomic step */
Boolean testAndSet(Boolean* lockvar);

/* concatenate source 1 and source2 into the string dest */
/* dest needs to bee big enough to contain both */
void stringConcat(char *dest, char *source1, char *source2);

/* translate a string to an integer */
void int2string(char *str, int i);

/* translate a string to a referenced integer */
int atoi(int* intValue, char* string);

/* translate an integer to a string */
void itoa(int n, char s[]);

/* call stringLength; call ioWrite */
void display( ThreadId tty, char* s);

/* string comparison */
int stringCompare(char* a, char* b);
```

6 Übersicht der Systemcalls: Syscall.h

Eine vollständige Beschreibung der Systemcalls finden Sie im Topsy Manual: im Anhang B, Kapitel 2.6 als Zusammenfassung, und in den einzelnen Kapiteln mit Erläuterungen: Kapitel 3.1 (Threads), Kapitel 4.2 (Memory), Kapitel 5.1 (I/O und Drivers).

Ferner ist der gesamte Sourcecode von Topsy (inklusive Systemcalls) auf dem WWW verfügbar:

`http://www.tik.ee.ethz.ch/~topsy/Source/files.html`

Liste der Systemcalls:

```
/* memory management */

SyscallError  vmAlloc(Address *addressPtr,
  unsigned long int  size);
SyscallError  vmFree(Address address);
SyscallError  vmMove(Address *addressPtr, ThreadId newOwner);
SyscallError  vmProtect(Address  startAdr, unsigned long int
size,
  ProtectionMode  pmode);
SyscallError  vmCleanup(ThreadId threadId);


/* thread management */

SyscallError tmMsgSend( ThreadId to, Message *msg);
SyscallError tmMsgRecv( ThreadId* from, MessageId msgId,
  Message* msg, int timeOut);
SyscallError tmStart( ThreadId* id, ThreadMainFunction
  mainFunction, ThreadArg parameter, char *name);
SyscallError tmKill(ThreadId id);
void tmExit();
void tmYield();
SyscallError tmGetInfo(ThreadId about, ThreadId* tid,
  ThreadId* ptid);
SyscallError tmGetFirst(ThreadInfo* info);
SyscallError tmGetNext(ThreadInfo* info);


/* io interface */

SyscallError ioOpen(int deviceNumber, ThreadId* id);
SyscallError ioClose(ThreadId id);
SyscallError ioRead(ThreadId id, char* buffer,
  unsigned long int* nOfBytes);
SyscallError ioWrite(ThreadId id, char* buffer,
  unsigned long int* nOfBytes);
SyscallError ioInit(ThreadId id);
```