Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Übungsblatt 1

Übung Betriebssysteme
WS 2007/2008

Dipl.-Inf. Bernhard Rabe
Betriebssysteme & Middleware

Inhalt

Organisation

Make

Fehlerbehandlung

Übungsaufgaben

- Gruppen von 2-3(4) Studenten
- zweiwöchentliche Ausgabe von Übungen
- Zulassung zur Klausur wenn bei **jeder** Aufgabenserie **mindestens 50 %** der Punkte erreicht wurden

Abgabe von Übungsaufgaben

Abgabesystem

- https://www.dcl.hpi.uni-potsdam.de/bsprak/
- Formatanweisungen auf dem Übungsblatt
- 1 Lösung pro Gruppe und Aufgabenserie
- Abgabe mindestens 24H vor dem Tutoriumstermin!

Demo

Kontrolle der Übungsaufgaben

- Mündliches Testat mit der **gesamten** Gruppe zum Tutoriumstermin
- Quelltexte werden entsprechend den Anweisungen auf dem Übungsblatt im Abgabesystem abgelegt
- Abgabezeiten beachten!

Tutoren

- Florian Reinhart
- Philip Maschke
- Dandy Fenz
- Sebastian Wätzoldt
- Stefan Liske
- Nicolas Rüger
- Sebastian Hübner

Hinweise zum Abgabesystem

- Vor den Absenden nochmals Auspacken ausprobieren
- Keine Umlaute (ä, ö, ü, Ä, Ö, Ü,...), Leerzeichen oder Sonderzeichen (ß, @, € ..) in Dateinamen

Tutoriumstermine

Termin gilt **nur** für das **1. Testat**

Weitere Termine in individuelle Absprache mit dem Tutor

Treffpunkt für 1. Testat ??

Referenz-Compiler

Windows

- Visual Studio 2005

cl.exe

Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 14.00.50727.42 for 80x86

- unterstützt kein C99 !

Linux

- gcc (GCC) >=3.3.5 (C-Compiler)

Referenz make

Windows

- nmake
- `Microsoft (R) Program Maintenance Utility Version 8.00.50727.762`

Linux

- GNU Make
- `make --version =>  GNU Make 3.81`

make

Regel basiertes Softwarewerkzeug

Zeitstempelunterschiede als Bedingung für die Anwendung von Regeln

Regeln

- Ziel
- Abhängigkeiten
- Aktionen

make

Regelpriorität

1. Übergabeparameter
2. Explizite (Makefile) Regeln
3. Implizite Regeln
   - make –p

Variablen

Demo

nmake

nmake unter Windows (→vsvars32.bat)

- `"%VS80COMNTOOLS%vsvars32.bat"`

Makefile oder **makefile**

MSDN

GNU make

GNU Make

- GNUmakefile -> **makefile** -> **Makefile**
- http://www.gnu.org/software/make/manual/make.html

Regeln

targets : *prerequisites*

<TAB>command

<TAB>...

Bsp.

```
hello.exe: hello.c
                    cl hello.c
```

Jedes Kommando wird in einer eigenen Shell ausgeführt

Regeln

Auswahl einer Regel beim Aufruf

- `nmake hello.exe`

Ausführung der 1. Regel anderenfalls

Bsp.:

```
all: hello.exe

hello.exe: hello.c
     cl hello.c
```

Variablen

Zuweisung

- OBJS=hello.obj main.obj

Verwendung

```
hello.exe:$(OBJS)
```

Variablen

CC C-Compiler

CFLAGS C-Compiler Optionen z.B. `-Wall`

LDLIBS Linker Bibliotheken

MAKE make Werkzeug

Bsp.:

```
hello.exe: hello.c
  $(CC) hello.c
```

Automatische Variablen

$@ Ziel

$< Name der 1. Abhängigkeit

$* Ziel ohne Endung

Bsp.:

```
hello.exe: hello.c
	$(CC)	$< -o bin\$@
=>		hello.c –o bin\hello.exe
```

make Parameter

-f Makefile

- ein spezielles Makefile benutzen

-n

- gibt alle auszuführenden Kommandos aus

VAR=Wert

- eine Variable mit einem Wert belegen
- `nmake CC=gcc`

hello.c PARAM

Fehlerbehandlung

API-Ruf konnte nicht „erfolgreich“ ausgeführt werden

Rückgabewert zeigt Status bzw. Fehlercode an

Ermitteln der Fehlerbeschreibung aus dem Fehlercode

Programmierrichtlinien

API

Windows API

C-Bibliothek

- POSIX-API
- http://www.opengroup.org/onlinepubs/007908799/headix.html

Windows API

Fehlerindikator unterschiedlich

- BOOL → Null bzw. FALSE
  - CreateProcess(..)
- HANDLE → NULL, INVALID_HANDLE_VALUE
  - CreateThread(..) NULL
  - CreateFile(..) INVALID_HANDLE_VALUE
- DWORD → MSDN
  - WaitForSingleObject(..) Status bzw. WAIT_FAILED

Windows API

Fehlercodes nach Kategorie

- System Error Codes (0-499)
- System Error Codes (500-999)
- System Error Codes (1000-1299)
- System Error Codes (1300-1699)
- System Error Codes (1700-3999)
- System Error Codes (4000-5999)
- System Error Codes (6000-8199)
- System Error Codes (8200-8999)
- System Error Codes (9000-11999)
- System Error Codes (12000-15999)

Windows API

DWORD GetLastError(**void**)

- liefert den Fehlercode des letzten Windows API Rufs durch den rufenden Thread, der einen Fehler liefert → MSDN

DWORD FormatMessage**(..)**

- Formatiert eine Nachricht, z.B. aus der Systemtabelle → Fehlercode
- Funktioniert nur für Windows API !

C-Bibliothek

Fehlercode Konstanten errno.h

```
#define EPERM           1
#define ENOENT          2
#define ESRCH           3
#define EINTR           4
#define EIO             5
#define ENXIO           6
#define E2BIG           7
#define ENOEXEC         8
#define EBADF           9
#define ECHILD          10
#define EAGAIN          11
...
```

C-Bibliothek

LibC-API Rufe liefern in der Regel <0 im Fehlerfall

- `open(), fork()`
- `malloc` → NULL

Fehlercode des letzten Rufs wird in der Variable **errno** gespeichert (errno.h)

- LibC-API Verhalten ist nicht konsistent
- `sem_init()` setzt errno
- `pthread_create()` liefert Fehlernummer, sonst 0

C-Bibliothek

```
void perror( const char *string )
```

- “string: Fehlermeldung” nach stderr
- “Fehlermeldung” wenn string==NULL
- LC_MESSAGES bestimmt die Sprache

C-Bibliothek

char ***strerror(int** *errnum* **)**

- Liefert einen Zeiger auf die Systemfehlermeldung zu **errnum** zurück
- perror Verhalten

```
#include <errno.h>
if(-1 == fork() )
{

        close(f);
        perror(“Error at fork”);
        exit(1);
}
```

C-Bibliothek

char *strerror(**int** *errnum*)

- Liefert einen Zeiger auf die Systemfehlermeldung zu **errnum** zurück
- perror Verhalten

```
#include <errno.h>
if(-1 == fork() )
{
        int errval=errno;
        close(f);
        fprintf(stderr,"%s: %s\n",string,strerror(errval));
        exit(1);
}
```