HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Präsentation der Bachelorarbeit

Christian Ritzler

Testanwendungen zur Überprüfung des PEARL-Systems auf Sprachkonformität

Erstbetreuer: Prof. Dr. Rainer Müller

Zweitbetreuer: Prof. Hannelore Frank

Agenda

- Motivation
- Aufgabenstellung
- Grundlagen
- Analyse
- Testanwendungen
- Ergebnis
- Ausblick

Motivation

- Projekt OpenPearl
  - PEARL-System (Compiler and Runtime System)
  - Opensource Projekt auf Sourceforge[1]
  - Leitung: Prof. Dr. Rainer Müller (HFU)

    Dipl.-Inform. Marcel Schaible (FernUniversität Hagen)
  - Ziele:
    - PEARL als Realtime Programmiersprache in der Lehre einzusetzen
    - Zielplattformen sind alle Linuxdistributionen und FreeRTOS
      - Einschränkung: C++ muss verfügbar sein

1. Sourceforgen.net ist eine Website, die es Programmierern ermöglicht quell-offene Projekte zu erstellen und zu verwalten.

Agenda

- Motivation
- **Aufgabenstellung**
- Grundlagen
- Analyse
- Testanwendungen
- Ergebnis
- Ausblick

Aufgabenstellung

- Ziele:
  - Entwicklung von Testprogrammen
    - Basis: DIN 66253-2 Spezifikation
  - Automatisierte starten und auswerten der Testprogramme
  - Abgrenzung:
    - Tasking, Semaphoren und Operatoren (FLOAT)
    - Plattform Linux (Debian 7)
- Weiteres Ziel:
  - Ergänzung der DIN 66253-2 Spezifikation

Agenda

- Motivation
- Aufgabenstellung
- **Grundlagen**
  - **Programmiersprache PEARL**
  - PEARL-Sprachsystem
- Analyse
- Testanwendungen
- Ergebnis
- Ausblick

Grundlagen
Programmiersprache PEARL

HOCHSCHULE FURTWANGEN UNIVERSITY HFU

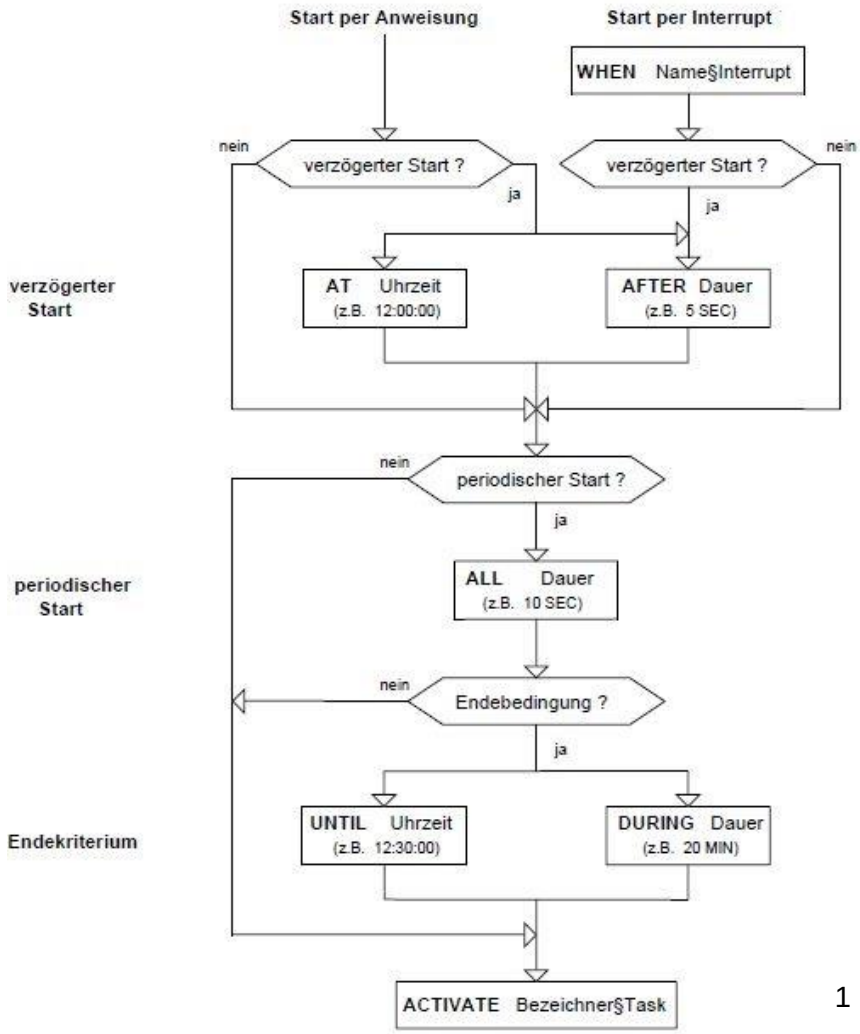
- Eckdaten:
  - Process and Experiment Automation Realtime Language (PEARL)
  - Höhere Programmiersprache (PASCAL artig)
  - Entstand 1975 an der Leibniz Universität in Hannover
  - Bietet eine komfortable u. sichere Programmierung von Multitasking- u. Echtzeitaufgaben
  - Aktuelle Normung ist die DIN 66253-2 (April 1998)

Grundlagen
Programmiersprache PEARL

- Sprachliche Mittel zur Tasksteuerung
  - ACTIVATE (Starten)
  - TERMINATE (Beenden)
  - SUSPEND (Anhalten)
  - CONTINUE (Fortsetzen)
  - RESUME (Verzögern)
  - PREVENT (Ausplanen)

Grundlagen
Programmiersprache PEARL

- Möglichkeiten eine Task direkt oder unter verschiedenen Startbedingungen zu starten



1

1. Quelle: GI-Fachgruppe 4.4.2 Echtzeitprogrammierung, PEARL 90 Sprachreport. Version 2.0. GI Gesellschaft für Informatik e.V., Bonn, Januar 1995, Seite 9-9

Grundlagen
Programmiersprache PEARL

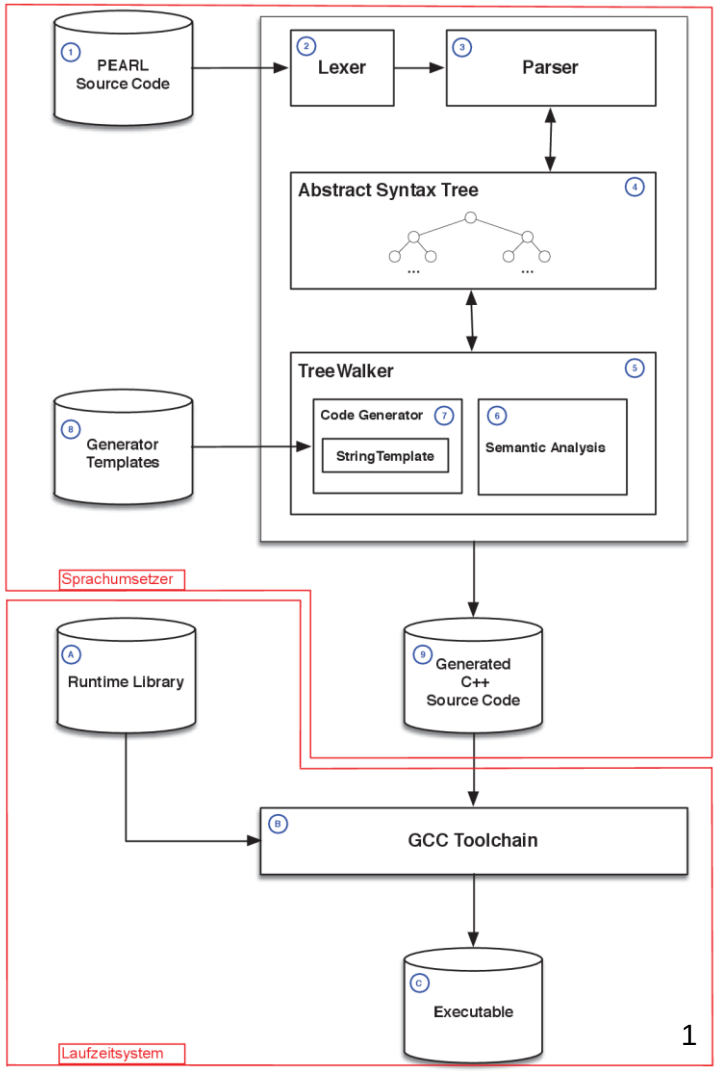
- Beispiele:
  - AT 12:00:00 ACTIVATE Protokoll;
  - AFTER 1 SEC ALL 2 SEC DURING 6 SEC ACTIVATE Protokoll;
  - AFTER 1.5 SEC RESUME TaskMain;
  - SUSPEND Task2;
  - CONTINUE Task2 PRIO 2;
  - PREVENT Protokoll;
  - TERMINATE Task2;

Agenda

- Motivation
- Aufgabenstellung
- **Grundlagen**
  - Programmiersprache PEARL
  - **PEARL-Sprachsystem**
- Analyse
- Testanwendungen
- Ergebnis
- Ausblick

Grundlagen
PEARL-Sprachsystem

- Sprachumsetzer
- Laufzeitsystem



1. Quelle: http://sourceforge.net/projects/openpearl/files/Diagrams/

Christian Ritzler, Fakultät MME,
Studiengang ETI

Agenda

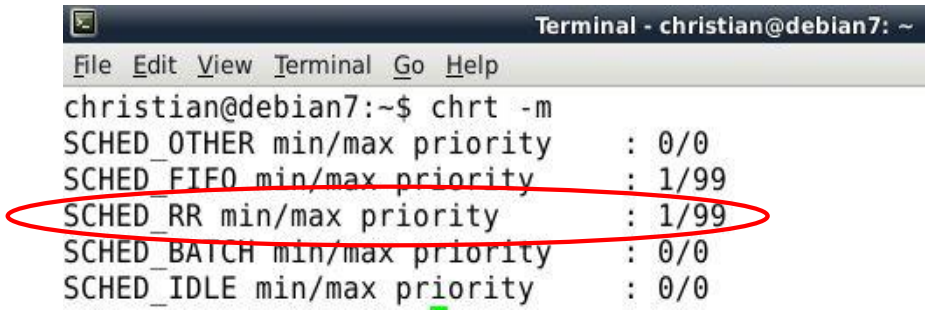
- Motivation
- Aufgabenstellung
- Grundlagen
- **Analyse**
- Testanwendungen
- Ergebnis
- Ausblick

Analyse
Scheduler in PEARL vergl. Linux

- PEARL
  - Preemptive Priority Scheduler
    +
  - Round-Robin-Prinzip
  - Priorität 1 … 255

- Linux User
  - Completely Fair Scheduler

- Linux Super-User



Analyse
Taskzustandstabelle (Scheduled Continue)

PEARL / Taskstate	Scheduled CONTINUE			executing Scheduled CONTINUE		
	SA	SC	TS	SA	SC	TS
active [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
active [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
active [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
active [sa = 1, sc = 1]	NC	1	NC	NC	0	NC
suspended [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
suspended [sa = 0, sc = 1]	NC	1	NC	NC	0	active
suspended [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
suspended [sa = 1, sc = 1]	NC	1	NC	NC	0	active
terminated [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
terminated [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
terminated [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
terminated [sa = 1, sc = 1]	NC	1	NC	NC	0	NC
blocked(synch) [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
blocked(synch) [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
blocked(synch) [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
blocked(synch) [sa = 1, sc = 1]	NC	1	NC	NC	0	NC
suspended(synch) [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
suspended(synch) [sa = 0, sc = 1]	NC	1	NC	NC	0	blocked (synch)
suspended(synch) [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
suspended(synch) [sa = 1, sc = 1]	NC	1	NC	NC	0	blocked (synch)
blocked(io) [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
blocked(io) [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
blocked(io) [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
blocked(io) [sa = 1, sc = 1]	NC	1	NC	NC	0	NC
suspended(io) [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
suspended(io) [sa = 0, sc = 1]	NC	1	NC	NC	0	blocked (io)
suspended(io) [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
suspended(io) [sa = 1, sc = 1]	NC	1	NC	NC	0	blocked (io)

Analyse
Taskzustandstabelle (Scheduled Continue)

Taskstate \ PEARL	Scheduled CONTINUE			executing Scheduled CONTINUE		
	SA	SC	TS	SA	SC	TS
active [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
active [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
active [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
active [sa = 1, sc = 1]	NC	1	NC	NC	0	NC
suspended [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
suspended [sa = 0, sc = 1]	NC	1	NC	NC	0	active
suspended [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
suspended [sa = 1, sc = 1]	NC	1	NC	NC	0	active
terminated [sa = 0, sc = 0]	NC	1	NC	NC	0	not possible
terminated [sa = 0, sc = 1]	NC	1	NC	NC	0	NC
terminated [sa = 1, sc = 0]	NC	1	NC	NC	0	not possible
terminated [sa = 1, sc = 1]	NC	1	NC	NC	0	NC

Taskstate \ PEARL	CONTINUE (by self)			CONTINUE (by other)		
	SA	SC	TS	SA	SC	TS
active [sa = 0, sc = 0]	NC	NC	NC	NC	NC	pot possible
active [sa = 0, sc = 1]	NC	0	NC	NC	NC	pot possible
active [sa = 1, sc = 0]	NC	NC	NC	NC	NC	pot possible
active [sa = 1, sc = 1]	NC	NC	NC	NC	NC	pot possible
suspended [sa = 0, sc = 0]	NC	NC	pot possible	NC	NC	active
suspended [sa = 0, sc = 1]	NC	NC	pot possible	NC	0	active
suspended [sa = 1, sc = 0]	NC	NC	pot possible	NC	NC	active
suspended [sa = 1, sc = 1]	NC	NC	pot possible	NC	0	active
terminated [sa = 0, sc = 0]	NC	NC	pot possible	NC	NC	NC + TaskTerminatedSignal
terminated [sa = 0, sc = 1]	NC	NC	pot possible	NC	0	NC + TaskTerminatedSignal
terminated [sa = 1, sc = 0]	NC	NC	pot possible	NC	NC	NC + TaskTerminatedSignal
terminated [sa = 1, sc = 1]	NC	NC	pot possible	NC	0	NC + TaskTerminatedSignal

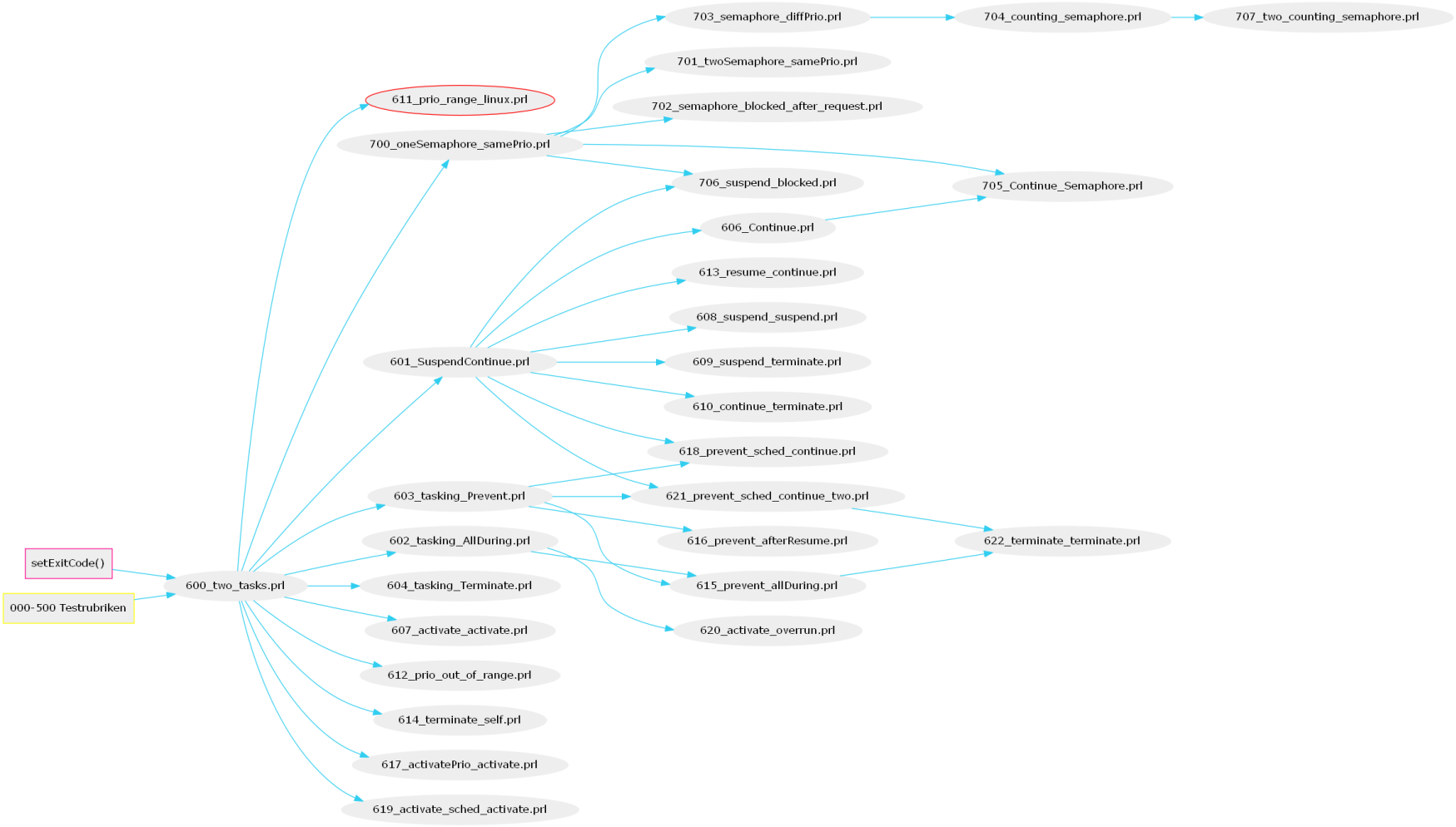
Agenda

- Motivation
- Aufgabenstellung
- Grundlagen
- Analyse
- **Testanwendungen**
- Ergebnis
- Ausblick

Testanwendungen
Beispiel zu Tasking

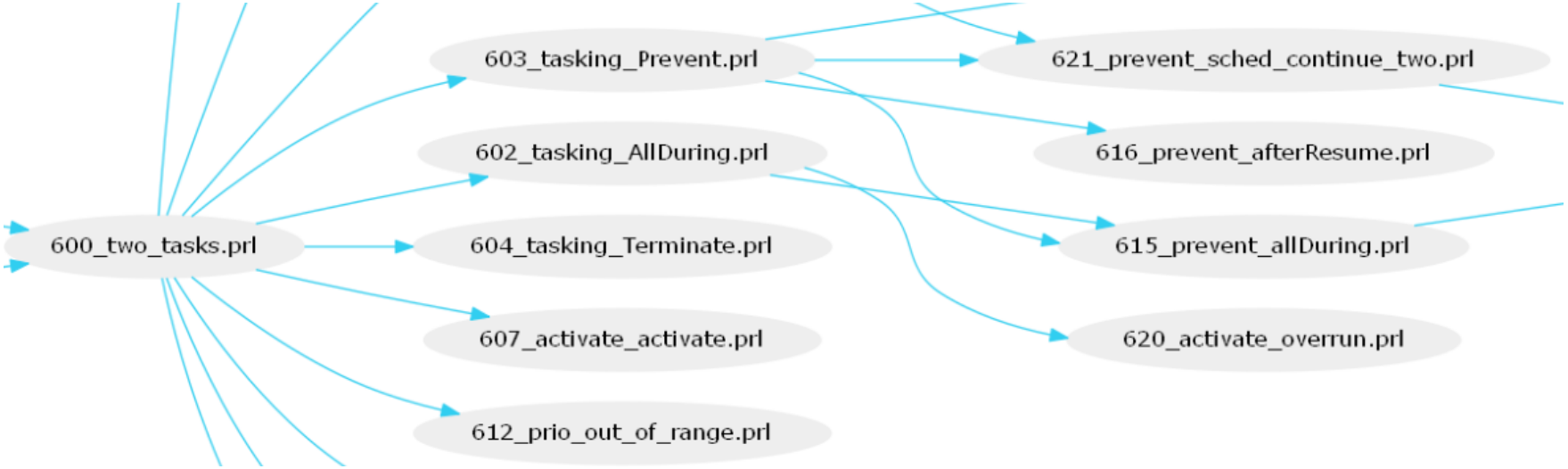
```
1.              …
2. Task1: TASK PRIO 3 MAIN;
3.      varTask1 := varTask1 + 1;
4.       ACTIVATE Task2 PRIO 2;
5.              ...
6.       ACTIVATE Task2;
7.              …
8.       AFTER 1 SEC RESUME;
9.       __cpp__('if(_varTask2.x == 2 && _varTask1.x == 1 && _check.x == 2) { ‘
10.                  '     pearlrt::Control::setExitCode(0); ‘
11.                              …
12. END;
13. Task2: TASK PRIO 4;
14.      varTask2 := varTask2 + 1;
15. END;
```

Testanwendungen
Abhängigkeiten Tasking



Christian Ritzler, Fakultät MME,
Studiengang ETI

Testanwendungen
Abhängigkeiten Tasking



Testanwendungen
Operatoren

- Datentyp FLOAT
  - FLOAT(24)
  - FLOAT(53)
- monadische u. dyadische Operatoren
- Operatoren Testprogramme prüfen nur die Funktionalität nicht die Genauigkeit
- Eine gewisse Ungenauigkeit wird toleriert
  - Epsilon bei FLOAT(24) = $1 * 10^{-6}$
  - Epsilon bei FLOAT(53) = $1 * 10^{-14}$

Testanwendungen
Automatisierte Starten und Auswerten

- 2 Shell Skripte
  - 1. Übersetzen und Starten
  - 2. Auswerten
- Shell Skripte sind im Makefile eingebunden
  - make build
  - make run

```
ritzlerc@pearl:~/pc/testsuite/run$ make help
make help      : this output
make build     : build all tests
make allrun    : tests for run results (all targets)
make clean     : remove all output files
make run       : tests for run results (except known problems)
make all       : build and tests all files
```

```
ritzlerc@pearl:~/pc/testsuite/run$ make build
try to build test programs
--------------------------
program                                 :    prl->c++      prl->bin
400_float_operators_one.prl             :     PASSED        PASSED
401_float_operators_two.prl             :     PASSED        PASSED
402_float_signal_command_length.prl     :     PASSED        PASSED
403_float_operators_three.prl           :     PASSED        PASSED
600_two_tasks.prl                       :     PASSED        PASSED
601_tasking_SuspendContinue.prl         :     PASSED        PASSED
602_tasking_AllDuring.prl               :     PASSED        PASSED
603_tasking_Prevent.prl                 :     PASSED        PASSED
604_tasking_Terminate.prl               :     PASSED        PASSED
605_tasking_activate_signal_one.prl     :     PASSED        PASSED
606_Continue.prl                        :     PASSED        PASSED
607_activate_activate.prl               :     PASSED        PASSED
608 suspend suspend.prl                 :     PASSED        PASSED
```

Testanwendungen
Automatisierte Starten und Auswerten

```
ritzlerc@pearl:~/pc/testsuite/run$ make run
run simple test programs
------------------------
400_float_operators_one                  :     pass
401_float_operators_two                  :     pass
402_float_signal_command_length          :     pass
403_float_operators_three                :     pass
600_two_tasks                            : *** fail ***
601_tasking_SuspendContinue              :     pass
602_tasking_AllDuring                    :     pass
603_tasking_Prevent                      :     pass
604_tasking_Terminate                    :     pass
605_tasking_activate_signal_one          :     pass
606_Continue                             :     pass
607_activate_activate                    :     pass
608_suspend_suspend                      :     pass
609_suspend_terminate                    :     pass
611_prio_range_linux                     : *** fail ***
612_prio_out_of_range                    :     pass
613_resume_continue                      :     pass
614_terminate_self                       :     pass
615_prevent_allDuring                    :     pass
616_prevent_afterResume                  :     pass
617_activatePrio_activate                : *** fail ***
618_prevent_sched_continue               :     pass
619_activate_sched_activate              :     pass
620_activate_overrun                     :     pass
621_prevent_sched_continue_two           :     pass
622_terminate_terminate                  :     pass
700_oneSemaphore_samePrio                : *** fail ***
701_twoSemaphore_samePrio                : *** fail ***
702_semaphore_blocked_after_request      : *** fail ***
703_semaphore_diffPrio                   :     pass
704_counting_semaphore                   : *** fail ***
705_Continue_Semaphore                   : *** fail ***
706_suspend_blocked                      :     pass
707_two_counting_semaphore               : *** fail ***
```

```
root@pearl:/home/ritzlerc/pc/testsuite/run# make run
run simple test programs
------------------------
400_float_operators_one                  :     pass
401_float_operators_two                  :     pass
402_float_signal_command_length          :     pass
403_float_operators_three                :     pass
600_two_tasks                            :     pass
601_tasking_SuspendContinue              :     pass
602_tasking_AllDuring                    :     pass
603_tasking_Prevent                      :     pass
604_tasking_Terminate                    :     pass
605_tasking_activate_signal_one          :     pass
606_Continue                             :     pass
607_activate_activate                    :     pass
608_suspend_suspend                      :     pass
609_suspend_terminate                    :     pass
611_prio_range_linux                     :     pass
612_prio_out_of_range                    :     pass
613_resume_continue                      :     pass
614_terminate_self                       :     pass
615_prevent_allDuring                    :     pass
616_prevent_afterResume                  :     pass
617_activatePrio_activate                :     pass
618_prevent_sched_continue               :     pass
619_activate_sched_activate              :     pass
620_activate_overrun                     :     pass
621_prevent_sched_continue_two           :     pass
622_terminate_terminate                  :     pass
700_oneSemaphore_samePrio                :     pass
701_twoSemaphore_samePrio                :     pass
702_semaphore_blocked_after_request      :     pass
703_semaphore_diffPrio                   :     pass
704_counting_semaphore                   :     pass
705_Continue_Semaphore                   :     pass
706_suspend_blocked                      :     pass
707_two_counting_semaphore               :     pass
```

Agenda

- Motivation
- Aufgabenstellung
- Grundlagen
- Analyse
- Testanwendungen
- **Ergebnis**
- Ausblick

Ergebnis

- 35 Fehler im PEARL-Sprachsystem entdeckt
  - 25 größere Fehler
  - 10 kleine Fehler
- Alle aufgedeckten Fehler sind behoben
- 2 Shell-Skripte für das automatisierte Starten und Auswerten
- Taskzustandstabellen

Agenda

- Motivation
- Aufgabenstellung
- Grundlagen
- Analyse
- Testanwendungen
- Ergebnis
- **Ausblick**

Ausblick

- Weitere Testprogramme
  - z.B. Ein- und Ausgabe
- Alle C++ Anteile aus den Testprogrammen entfernen
- Belastungs- und Genauigkeitstests
- Portierung der Testprogramme auf eine andere Plattform

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!

Name: Christian Ritzler
Email: christian.ritzler@hs-furtwangen.de

Weitere Infos unter:
http://sourceforge.net/projects/openpearl/?source=navbar