

Vertiefung Systemtechnik

Prof. Dr. Peter Gerwinski

15. Oktober 2012

2 Theorie der Echtzeitsysteme

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

2 Theorie der Echtzeitsysteme

2.1 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen

2 Theorie der Echtzeitsysteme

2.1 Multitasking

- *Kooperatives Multitasking*

Prozesse geben freiwillig Rechenzeit ab

- *Präemptives Multitasking*

Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)

- *Scheduler*

Steuerprogramm, das Prozessen Rechenzeit zuteilt

- *Kontextwechsel*

Umschalten zwischen zwei Prozessen

- Das Betriebssystem wurde per Timer-Interrupt aufgerufen.
- Registerinhalte des unterbrochenen Prozesses liegen auf dem CPU-Stack
- Registerinhalte des neuen Prozesses auf den CPU-Stack legen
- Rücksprung nicht zum unterbrochenen, sondern in den neuen Prozeß

→ Stack austauschen

2 Theorie der Echtzeitsysteme

2.1 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen
- *Round-Robin-Verfahren (Rundlauf)*
Zuteilung von *Zeitschlitz*en auf einer *Zeitscheibe* an die Prozesse

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt:
Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt: Zähler des aktuellen Tasks wird dekrementiert; Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt: Zähler des aktuellen Tasks wird dekrementiert; Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.
- *keine* Echtzeit

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm

Linux 0.01

- Timer-Interrupt: Zähler des aktuellen Tasks wird dekrementiert; Task mit höchstem Zähler bekommt Rechenzeit.
- Wenn es keinen lafbereiten Task mit positivem Zähler gibt, bekommen alle Tasks gemäß ihrer Priorität neue Zähler zugewiesen.
- *keine* harte Echtzeit

Zombies

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

- Warum? —> Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“),
bekommt keine Rechenzeit mehr („seiner Seele beraubt“),
hat alle belegten Ressourcen wieder freigegeben („willenlos“),
wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? → Nein.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“), bekommt keine Rechenzeit mehr („seiner Seele beraubt“), hat alle belegten Ressourcen wieder freigegeben („willenlos“), wird aber noch in der Prozeßliste geführt („untot“).

- Warum? —> Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? —> Nein.
- Aber? —> Wenn sich Zombie-Prozesse anhäufen, deutet dies auf einen Prozeß hin, der andere Prozesse erzeugt und anschließend „hängt“.

Zombies

Wikipedia:

Als Zombie wird die fiktive Figur eines zum Leben erweckten Toten (Untoter) oder eines seiner Seele beraubten, willenlosen Wesens bezeichnet. Der Begriff leitet sich von dem Wort nzùmbe aus der zentralafrikanischen Sprache Kimbundu ab und bezeichnet dort ursprünglich einen Totengeist.

Ein Zombie-Prozeß ist bereits beendet („tot“), bekommt keine Rechenzeit mehr („seiner Seele beraubt“), hat alle belegten Ressourcen wieder freigegeben („willenlos“), wird aber noch in der Prozeßliste geführt („untot“).

- Warum? → Ein anderer Prozeß (Elternprozeß) wartet noch auf den Rückgabewert des beendeten Prozesses.
- Schadet das? → Nein.
- Aber? → Wenn sich Zombie-Prozesse anhäufen, deutet dies auf einen Prozeß hin, der andere Prozesse erzeugt und anschließend „hängt“.
- Beispiel: Datenträger entfernt, zugreifender Prozeß „hängt“.
→ Tochterprozesse werden zu Zombies.

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich ausführlich

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich ausführlich
- Verschiedene Methoden der Priorisierung

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste* vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich ausführlich
- Verschiedene Methoden
der Priorisierung
→ später

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits (IPv4) bzw. Traffic-Class-Bits (IPv6) im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Qualitätssicherung beim Multitasking

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Durchsatz*
→ kommt gleich kurz dran
- Ressourcen reservieren:
Mutexe (= spezielle *Semaphore*)
→ kommt gleich ausführlich
- Verschiedene Methoden
der Priorisierung
→ später
- Umgehung der Probleme durch
speziell geschriebene Software
→ s. o. (MultiWii, RP6, ...)

Qualitätssicherung für Netzwerke:

- Verschiedene Anforderungen:
Latenz vs. *Jitter* vs. *Verluste*
vs. *Durchsatz*
- Ressourcen reservieren:
IntServ mit *Resource Reservation Protocol (RSVP)*
- Klassifizierung und Priorisierung:
DiffServ mit Type-of-Service-Bits
(IPv4) bzw. Traffic-Class-Bits (IPv6)
im IP-Header
- Eigenes Protokoll (Telefondienste):
Asynchronous Transfer Mode (ATM)

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Verschiedene Anforderungen

- *Latenz*: interaktive Anwendungen
- *Jitter*: Echtzeitanwendungen
- *Durchsatz*: Stapelverarbeitung

2 Theorie der Echtzeitsysteme

2.2 Ressourcen

Ressourcen reservieren

- *Semaphor*
griechisch: *sema* – Zeichen, *pherein* – tragen
„Eisenbahnsignal“
- *Mutex*
englisch: *mutual exclusion* – wechselseitiger Ausschluß
- *Spinlock*
englisch: *spin* – rotieren, *lock* Sperre