Kapitel 10: Prioritätsinversion

- Motivation und Basisbegriffe
- Betriebsmittelstau mit Prioritätsumkehr
- Betriebsmittelzuordnungsprotokolle
    - Nicht präemptive kritische Abschnitte
    - Prioritätsvererbung
    - Prioritätsceiling Protokolle

10.1 Motivation und Einführung

Die im Folgenden behandelten Betriebsmittelzuordnungsprotokolle spielen im Rahmen von Echtzeitanwendungen eine große Rolle, denn in diesem Anwendungsgebiet wird häufig mit einer prioritätsorientierter CPU-Schedulingstrategie gearbeitet. Im Folgenden werden nur Verfahren betrachtet, die in einem Einprozessorsystem angewendet werden können, wobei die zusätzlich ins Kalkül zu ziehenden exklusiven Betriebsmittel (*resources*), von den Prozessen dynamisch angefordert werden dürfen.

10.2 Ressourcenstau mit Prioritätsumkehr

Unter ungünstigen Zeitbedingungen kann es zu einer so genannten Prioritätsumkehr (*priority inversion*) kommen, wenn ein niederpriorer Prozess eine Ressource erhält, die später von einem hochprioren Prozess ebenfalls benötigt wird. Da die Ressource nicht entziehbar sein soll, muss der hochpriore Prozes auf das Freiwerden dieser Ressource warten und wird deswegen blockiert. Dies ist zwar ein wenig erfreuliches Szenario, stellt aber noch nicht den Fall der Prioritätsumkehr dar.

Erst wenn nun noch ein Prozess mittlerer Priorität hinzukommt, dann wird die Wartezeit des hochpriorren Prozesses weiter unnötig verlängert, denn der niederpriore Prozess, der die Ressource wieder freigeben könnte, kommt ja wegen des Prioritätenschemas nicht an die Reihe. Dieses Verhalten konnte sehr einprägsam am Marspathfinder in den neunziger Jahren beobachtet werden. Obwohl das RTOS schon damals zur Vermeidung von Prioritätsumkehr optional das Prioritätsvererbungsprotokoll angeboten hatte, wurde diese Option von Anwendungsprogrammiererteam nicht verwendet mit den bekannten Auswirkungen, dass das Marsmobil, das durch drei Anwendungsprozesse kontrolliert wurde, tagelang vor dem "Felsen" Yogi festhing.

Die Crux an dieser Situation liegt darin, dass man stur am starren Prioritätenchema festhält, obwohl man leicht feststellen könnte, weswegen der hochpriore Prozess aufgehalten wird. Man spricht in diesem Zusammenhang davon, dass man diese Wartezeit der hochprioren Prozesse möglichst klein halten möchte. Die folgenden Ressourcenzuodnungrotokolle sind dem Buch von Liu: Real-Time Systems entnommen.

10.3 Ressourcenzuordnungsprotokolle

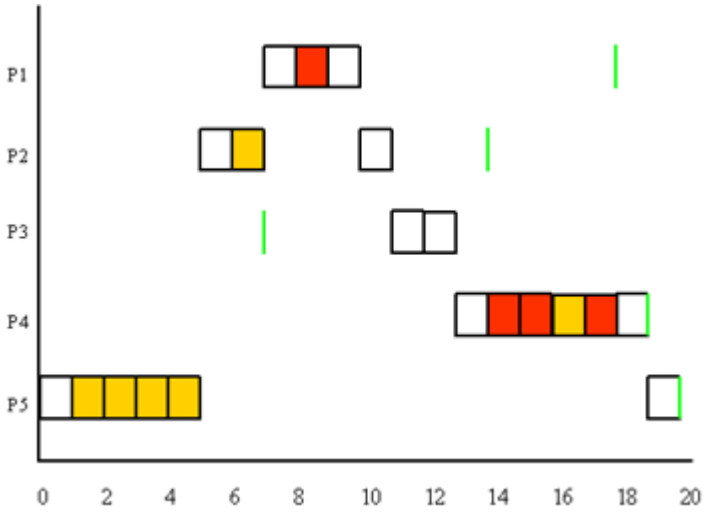
10.3.1 Nicht präemptive kritische Abschnitte

Ein sehr einfach zu implementierendes Protokoll sieht vor, dass der Prozess, der als erster irgendeine exklusive Ressource anfordert, solange zum wichtigsten Prozess im System wird, bis er diese Ressource nicht mehr benötigt, so dass es also keinen Grund gibt, diesen Prozess innerhalb seines kritischen Abschnitts (der Ressourcennutzung) vom Prozessor zu verdrängen. Jede weitere Ressourcennutzung dieses Prozesses kann demnach sofort befriedigt werden, da dieser Prozess quasi konkurrenzlos ist.

Prinzipiell garantiert dieses Protokoll, dass die Prioritätsumkehrphase möglichst kurz ist, allerdings reduziert dieses Protokoll erheblich mögliche Nebenläufigkeiten im System, mit solchen Prozessen, die nie in Konflikt

mit diesem Prozess treten würden. Außerdem garantiert das Protokoll Verklemmungsfreiheit, da wie gesagt, um die Ressourcen gar kein Konflikt entstehen kann. Das Protokoll verlangt zudem eine weitere Einschränkung, denn wenn während der Nutzungsphase einer Ressource ein blockierender Systemaufruf (system call) aufgerufen wird, darf gleichwohl der Prozessor keinem anderen Thread zugeordnet werden, was dieses Protokoll in der Praxis letztlich unbrauchbar macht.

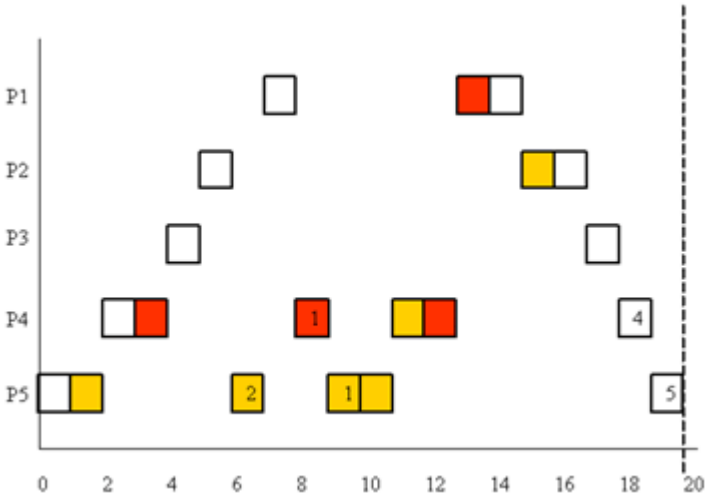
Im folgenden Bild ist das Ergebnis des NPCR Protokolls im Vergleich zum reinen CPU-orientierten Protokoll (starres Prioritätenschema mit Verdrängung) aufgezeigt. Wie man leicht sieht, ist die Dauer der Prioritätsumkehr bei den beiden wichtigeren Prozessen P1 und P2 deutlich verbessert.



Man beachte hierbei, dass der Prozess P4 im Verlauf seiner Abarbeitung zuerst das Betriebsmittel R1 und zwei Zeiteinheiten später zusätzlich noch das Betriebsmittel R2 anfordert; es liegt also der Fall hierarchischer Betriebsmittelanforderungen (*nested critical section*) vor.

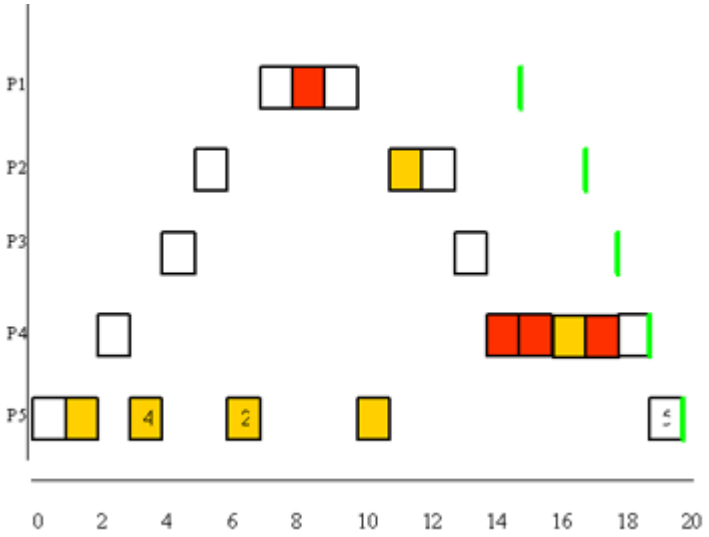
10.3.2 Prioritätsvererbung

Beim nächsten Ressourcenzuordnungsprotokoll erbt ein unwichtiger, aktuell die Ressource haltender Prozess, die höchste Priorität eines wegen ihm wartenden Prozesses, sofern diese höchste Priorität höher als die aktuelle Prozesspriorität ist. Dieses Protokoll reduziert ebenfalls die Prioritätsumkehrdauer, aber anders als das NPCS-Protokoll kann es keine Verklemmungsfreiheit garantieren. Beiden Protokollen gemeinsam ist, dass sie keinerlei Vorkenntnisse über den zukünftigen Ressourcenbedarf der beteiligten Prozesse haben müssen.



10.3.3 Prioritätsceiling Protokolle

Bei diesen Protokollen muss das System wissen, welche Ressourcen jeder Prozess im Zuge seiner Abarbeitung benötigen wird. Bei diesem Verfahren wird pro Ressource eine "Hürde" (*ceiling*) bestimmt, die gleich der Priorität des höchstprioren Prozesses ist, der diese Ressource anfordern wird. Ein Prozess, der eine gerade freie Ressource belegen will, muss eine höhere Priorität besitzen, als der aktuelle Systemhürdenwert ist, der wiederum gleich dem Maximum aller Ressourcenhürdenwerte ist. Auf diese Weise verhindert man, dass weniger wichtige Prozesse sich Ressourcen aneignen können, die später von den wichtigen Prozessen ebenfalls benötigt werden.



Das obige Bild zeigt das Ergebnis des Ceilingprotokolls in Relation zur Prioritätsvererbung. Der Nachteil dieses Verfahrens wie auch des Stackbased-Priority Ceiling Protokolls liegt darin, dass von allen Prozessen deren BM-Anforderungen zum Startzeitpunkt bekannt sein müssen.

Das ganze Vorgehen erinnert stark an die Verklemmungsvermeidung (*avoidance*), demzufolge sind beide Ceiling-Protokolle auch garantiert verklemmungsfrei.