NetSemaphore interface in Python and SAD

高エネルギー加速器研究機構
加速器研究部
小田切淳一、山本　昇
Jun-ichi Odagiri and Noboru Yamamoto
High Energy Accelerato Research Organization
Accelerator Laboratory
1-1 Oho, Tsukuba
Ibaraki, Japan

June 9, 2000

Abstract

ネットワーク・セマファ(Network semaphore) はネットワーク分散環境下での排他制御を実現するために著者の一人〔小田切〕によって開発された仕組みである。開発当初は C 言語による API だけが用意されていたが、今回スクリプト言語である Python および SAD からこのネットワーク・セマファを利用できるよう、これらの言語における API を開発した。

1 Introduction

実時間制御システムでは、多数のプロセスが同一のリソースに同時にアクセスするために生じる問題を回避するための仕組みとしてセマファ(Semaphore) がよく利用されている。[1]　実時間 OS 上のセマファは同一計算機上の複数のプロセスの競合を避けるためには有効であるが、KEKB 制御システムの様に、多数の計算機に分散された実時間制御システムではこのセマファを直接利用することはできない。一方、分散計算機環境下においても制御システム内の資源（例えば、BPM システムであるとか、Magnet PS 一式また、同一フィードバックアプリケーションの複数起動の禁止) を複数のプロセス間で排他的に制御する必要がある場合がある。制御システムは異種の計算機が共存する事を考慮すると〔実際 KEKB 制御システムでは HP-UX/True 64/VxWorks/Linux などの複数の OS と複数の計算機が共存している。）排他制御のための仕組みは OS independent なものであることが求められる。

Network Semaphore はこのような背景のもとに、著者の一人 (小田切) によって開発された。ネットワークセマファはネットワーク上に存在するネットワーク・セマファ・サーバに作られるセマファを、分散環境下のプロセスから取得 (Take)・解放 (Give) することで、プロセス間の排他制御を実現する。基本操作として、Take および Give が定義されている。Take は VxWorks でのセマファと同じように、セマファが既に他のプロセスによって取得されている場合の解放までの待ち時間を指定することができる。

Network Semaphore の実装の詳細については、文献 [1] を参照のこと。

2 SAD Interface

SAD から Network semaphore を利用するための二つの関数、`NetSemTake` および `NetSemGive` が定義されている。これらの関数は C 言語および Fortran 言語を用いて実装された。

[1] セマファ(semaphore) の元々の意味は鉄道などで使われた腕木信号器である。

2.1 NetSemTake

`NetSemTake[ servername:String, semid:Interger, timeout:Integer]`

`NetSemTake` はサーバ名が servername である network semaphore サーバから semid で指定されるセマファを取得する。取得に成功すると０、失敗すると-1 をその値として返す。セマファの ID, semid, は 0 から 255 までの値が許される。

既にセマファが他のプロセスによって取得されていると、timeout（０から 65535 の範囲が許される。）に指定されている秒数のあいだ、この関数の呼び出しプロセスの実行がブロックされる。timeout 秒経過まえに、セマファのオーナがセマファを返却した場合にはこの関数呼出は直ちに終了し、０を値として返す。timeout に０を指定した場合には、セマファを取得できない場合にも関数の実行は直ちに終了する。この場合値として-1 が返される。

同一のネットワーク・セマファ・サーバ上には、0 255 のセマファID をもつセマファが存在する。

同一プロセスから複数回 `NetSemTake` を呼び出す事ができる。この場合には、このプロセスが `NetSemTake` を呼び出したのと同じ回数 `NetSemGive` を呼び出して、セマファを完全に解放するまで、他のプロセスがこのセマファを取得することができない。

現在の実装では、servername の指定は最初の Network セマファ関数の呼び出しだけで意味を持っている。その後のこれらの関数呼出では、呼び出し時にこの変数を省略することはできないが、実際には使われない。これは、近い将来に複数の Network Semaphore　サーバが同一ネットワーク上で稼働する場合に備えて、API が定義されたためである。

2.2 NetSemGive

`NetSemGive[ servername:String, semid:Interger]`

`NetSemGive` は `NetSemTake` 関数呼び出しで取得した Network Semaphore を解放する。正常に解放された場合、0 を値として返す。セマファの所有者で無いプロセスで `NetSemGive` 関数を呼び出したばあいには、-1 を値として返す。

同一プロセス中で、複数回セマファを取得した場合には、その回数と同じ回数だけ `NetSemGive` を呼び出さないと、セマファはそのプロセスからは解放されない。

現在の実装では、servername の指定は最初の Network セマファ関数の呼び出しだけで意味を持っている。その後のこれらの関数呼出では、呼び出し時にこの変数を省略することはできないが、実際には使われない。これは、近い将来に複数の Network Semaphore　サーバが同一ネットワーク上で稼働する場合に備えて、API が定義されたためである

3 Python Interface

Python からこのネットワーク・セマファを利用するために netSemaphore モジュールが用意されている。netSemaphore モジュールを利用するには、まず

```
import netSemaphore
```

あるいは

```
from netSemaphore import *
```

を実行する。

3.1 netSemaphore.Take(server_name, semaphore_id, time_out_second)

SAD の `NetSemTake` を参照。

3.2 netSemaphore.Give(server_name, semaphore_id)

SAD の `NetSemTGive` を参照。

3.3 netSemaphore.Info(server_name, semaphore_id,info_switch)

ネットワーク・セマファ・サーバ、server_name、にたいして、semaphore_id で指定されるセマファの状態を問い合わせる。info_switch が 0 の場合には、現在のセマファの所有プロセスの ID、info_switch が 1 の場合には, そのセマファの取得待ち状態のプロセスの ID のリストを問い合わせる。戻り値は (code, info) の形式で、code は情報取得の成功時には 0、それ以外では-1 となる。また、info はプロセス ID を表す文字列で、”host 名:プロセス ID”の形式である。複数のプロセス ID がある場合には、info はそれらの ID を TAB コードを間に挟んで連結した文字列となる。

4 ネットワーク・セマファサーバの起動

ネットワーク・セマファサーバは VxWorks 上のアプリケーションである。ネットワーク・セマファの利用に先立ってこのサーバをいずれかの IOC で起動しておく必要がある。

ネットワーク・セマファ・サーバの実行可能形式のプログラム・ファイルは、

```
abco1:/cont/epics313/base/bin/pcore750
```

に置かれている。ネットワーク・セマファ・サーバの起動には、

```
-> ld < /cont/epics313/base/bin/pcore750/netsemServer.o
-> netSemServerInit
```

を VME 計算機のコンソール端末から入力してやるか、あるいはスタート・アップ　スクリプトに追加しておく。

5 ネットワーク・セマファの問題点あるいは今後の改良点

ネットワーク・セマファサーバは、セマファを取得しているアプリケーションがプログラムエラー等で終了した場合には、ソケット接続の切断を検出し、セマファを解放する。従って、このような場合にも、新たにクライアント・アプリケーションを起動するだけで、システムの動作は継続される。同一プロセス内で複数回セマファを取得したプロセスの場合であっても、このプロセスの異常終了時には、全てのセマファがサーバ側で解放される。

一方、セマファをクライアントが取得している状態で、 ネットワーク・セマファ・サーバあるいはネットワーク・セマファサーバが稼働している計算機自体が停止した場合、クライアントがこれを検出する機能はまだ提供されていない。　ネットワーク・セマファ・サーバを再起動すると、そのセマファは取得されていない状態から始まるため、ネットワーク上の二つのクライアント・プログラム (サーバ停止前に取得したクライアントとサーバ再起動後に取得したクライアント）が、セマファを取得した状態になってしまう可能性がある。　現状では、セマファ・サーバ停止時には、そのセマファ・サーバを利用している全てのクライアントプログラムの再起動 (あるいは、セマファの強制解放が必要である。

参考文献

[1] 小田切淳一, “ ソケット・インタフェースを用いたネットワーク・セマファ実装の試み”