複数プログラムを指定できるSTARTコマンド

2016年01月27日
指定するファイル名に空白が含まれていると実行できない場合があるので対処。

2014年12月9日
dummy.bat いちいち書くの面倒、と言うことなのでシンタックスシュガー dummyfile を追加。

2014年12月4日
必要に迫られ、でもジョブスケジューラを導入するのは大げさなのでつくっちゃった。

PARASTART.EXE ダウンロード

PARASTART.EXEのアーカイブ 2020年05月17日版

GitHub.comのリポジトリにもコードを公開しました。

See バッチ処理用ツール使い方例

PARASTARTコマンドの簡単な説明

WindowsのSTARTコマンドで指定のプログラムをバックグラウンドで実行させることができます。 しかしSTARTコマンドは同時に複数のプログラムを指定できません。

sample1.batではa.bat b.bat c.bat をバックグラウンドで起動し d.bat を起動しています。a.bat b.bat c.bat それぞれの実行終了を待たずに d.bat が実行されることになります。 もしa.bat b.bat c.bat の終了を待ってから d.bat を実行したい場合、STARTコマンドでは実現できません。

sample1.bat

```
@echo off
start a.bat
start b.bat
start c.bat
call d.bat
```

PARASTARTコマンドは、指定のプログラムを同時にバックグラウンドで実行し、それらが全て終了するまで待たせることができます。

sample2.batではa.bat b.bat c.bat をPARASTARTコマンドから同時に実行しています。また、PARASTARTコマンドは a.bat b.bat c.bat の全ての実行が終了してから自身を終了させるため、a.bat b.bat c.bat の終了を待って d.bat を実行させることができるようになります。

sample2.bat

```
@echo off
parastart a.bat b.bat c.bat
call d.bat
```

またはsample3.batのようにBGWAITコマンドを使って a.bat b.bat c.bat の終了を待たせる方法もあります。 ただしこの場合はPARASTARTコマンドの -wait オプションを使うような、終了待ち時間(実行時間制限)を指定することができません。

sample3.bat

```
@echo off
start a.bat
start b.bat
start c.bat
bgwait
call d.bat
```

PARASTARTコマンドの実行時条件

parastart.exe の利用には .NET Framework 4.0 の導入されたWindows環境が必要です。 バッチファイルプログラム(*.bat□*.cmd)のほか、コンソールアプリケーションであれば ROBOCOPY のようなプログラムも指定できます□※GUIをもつアプリケーションはプロセス終了を検出できません。

```
E:\>parastart
parastart.exe - Parallel execution of multiple programs.

usage: parastart.exe [-any|-all|-zero] [-wait n] cmd1 cmd2 cmd3 ...

-zero   : Default. Return code is always zero.
-any    : If one or more of the program has been successfully, the return
code is zero.
-all    : If all of the program has been successfully, the return code is
zero.
-wait n : Set n seconds to the execution limit time of each program. *
Default is unlimited.
cmd1 cmd2 cmd3... : List of *.bat or *.cmd program that execution in
parallel.

E:\>
```

-zero	cmd1 cmd2 cmd3 ...の各プログラムのリターンコードが何であってもPARASTARTコマンド自身のリターンコードに 0 を返します。 -any オプションも -all オプションも指定しなかった場合□ -zero オプションが指定されたことになります。

-any	cmd1 cmd2 cmd3 ...のいずれかのプログラムがリターンコード 0 で終了した場合にPARASTARTコマンド自身のリターンコードに 0 を返します。 それ以外は16を返します。
-all	cmd1 cmd2 cmd3 ...の全プログラムがリターンコード 0 で終了した場合にPARASTARTコマンド自身のリターンコードに 0 を返します。 それ以外は16を返します。
-wait n	cmd1 cmd2 cmd3 ...の全プログラムの実行時間を n 秒に制限します n 秒以内に終了しなければPARASTARTコマンドは全てのプログラムを強制的に終了します。 デフォルトは 0 秒で、このときは無制限を意味します。
cmd1 cmd2 cmd3 ...	*.bat か *.cmd の名称を持つバッチファイル。またはコンソールアプリケーションのプログラム PARASTARTコマンドの子プロセスとして実行されます。

オプションを複数指定した場合、最後に指定したものが適用されます。

たとえば、

```
parastart -any -wait 100 -all -wait 30 a.bat b.bat c.bat
parastart -wait 30 -any -zero -all a.bat b.bat c.bat
```

の指定はそれぞれ

```
parastart -all -wait 30 a.bat b.bat c.bat
parastart -wait 30 -all a.bat b.bat c.bat
```

を指定したことになります。

内部的には /C オプションをつけた CMD.EXE を経由してプログラムを実行させています。
たとえば、以下のようなコマンドラインの場合、

```
E:\>parastart a.bat "dir d:\ > d.txt" "ROBOCOPY D:\DATA \\NASSERVER\BACKUP
/MIR"
```

以下の3つがバックグラウンドで並列実行されます。

```
CMD.EXE /C a.bat
CMD.EXE /C dir d:\ > d.txt
CMD.EXE /C ROBOCOPY D:\DATA \\NASSERVER\BACKUP /MIR
```

引数指定時の制限

PARASTARTコマンドは、ハイフン(-)から始まる引数はオプションとみなします。そのため、先頭にハイフンのある引数を、バッチファイル名やプログラム名として指定できません。

この例では -a.bat -b.bat -c.bat はオプションとして解釈されます。ですが正しいオプションではないため無視され -a.bat -b.bat -c.bat は実行されません。

```
parastart -wait 180 -a.bat -b.bat -c.bat
```

どうしてもハイフンから始まる名称を持つバッチファイルやプログラムを実行したい場合は、以下のようにファイル名を書く直前に “dummyfile” を記述するか、リターンコードゼロを返すバッチファイ

ル(この例ではdummy.bat)を記述してください。

```
parastart -wait 180 dummyfile -a.bat -b.bat -c.bat
parastart -wait 180 dummy.bat -a.bat -b.bat -c.bat
```

前者の記述だと□dummyfile の後ろの -a.bat -b.bat -c.bat が実行されます。
後者の記述だと□dummy.bat -a.bat -b.bat -c.bat が実行されます。

PARASTARTコマンドが引数を解析する際、最初のバッチファイル名/プログラム名/dummyfile を見つけると、以降に続く引数は全てバッチファイル名/プログラム名だとみなして処理します。

dummyfile はハイフンから始まるファイル名を指定するために用意された糖衣構文用のダミーファイル名です。同じ名称のファイルがあっても実行しません。引数解析時に読み飛ばしされるだけです。

ソース

晒せと言われたんでさらしますがたいしたことはしていません。

Program.cs

```
using System;
using System.Collections;
using System.Text.RegularExpressions;
using System.Diagnostics;
using System.Threading.Tasks;

namespace parastart
{
    enum retcodeType
    {
        allways
       , anysafe
       , allsafe
    }

    class Program
    {

        static int Main(string[] args)
        {
            parsargs p = new parsargs(args);
            Process[] pa;
            Task[] ta;

            if (0 == p.commands.Length)
            {
                help();
                Environment.Exit(0);
            }
```

```
            pa = new Process[p.commands.Length];
            ta = new Task[p.commands.Length];

            for (int i = 0; i < p.commands.Length; i++)
            {
                int ii = i;

                pa[ii] = new Process();
                ta[ii] = new Task(() =>
                {
                    pa[ii].StartInfo.FileName = "CMD.EXE";
                    pa[ii].StartInfo.Arguments = "/C;" + "\"" +
p.commands[ii] + "\"";
                    pa[ii].StartInfo.UseShellExecute = false;
                    pa[ii].StartInfo.CreateNoWindow = true;
                    pa[ii].StartInfo.ErrorDialog = true;
                    pa[ii].Start();
                    if (0 != p.wait)
                    {
                        pa[ii].WaitForExit(p.wait);
                    }
                    else
                    {
                        pa[ii].WaitForExit();
                    }
                });

                ta[ii].Start();
            }

            Task.WaitAll(ta);

            int ret = 16;
            int safecount = 0;

            for (int i = 0; i < pa.Length; i++)
            {
                if (0 == pa[i].ExitCode) safecount++;
            }

            if ((p.runmode == retcodeType.allsafe) && (pa.Length ==
safecount))
            {
                ret = 0;
            }
            if ((p.runmode == retcodeType.anysafe) && (0 != safecount))
            {
                ret = 0;
            }

            return ret;
```

```
        }

        static void help()
        {
            String pn = "parastart.exe";
            Console.WriteLine("{0} - Parallel execution of multiple
programs.", pn);
            Console.WriteLine("\nusage: {0} [-any|-all|-zero] [-wait n]
cmd1 cmd2 cmd3 ...\n", pn);
            Console.WriteLine("-zero   : Default. Return code is always
zero.");
            Console.WriteLine("-any    : If one or more of the program
has been successfully, the return code is zero.");
            Console.WriteLine("-all    : If all of the program has been
successfully, the return code is zero.");
            Console.WriteLine("-wait n : Set n seconds to the execution
limit time of each program. * Default is unlimited.");
            Console.WriteLine("cmd1 cmd2 cmd3... : List of *.bat or
*.cmd program that execution in parallel.");
        }
    }

    class parsargs
    {
        String confFile = "";
        retcodeType retType = retcodeType.allways;
        int waitTime = 0;
        String[] cmds;

        Regex optReg = new Regex(@"^-[a-zA-Z]+");
        String dummyfile = "dummyfile";

        public retcodeType runmode
        {
            get
            {
                return retType;
            }
        }
        public String config
        {
            get
            {
                return confFile;
            }
        }
        public String[] commands
        {
            get
            {
                return cmds;
```

```
            }
        }
        public int wait
        {
            get
            {
                return waitTime;
            }
        }
        public parsargs(string[] args)
        {
            Boolean swcmdflg = false;
            ArrayList ag = new ArrayList();

            for (int i = 0; i < args.Length; i++)
            {
                if ((true == optReg.IsMatch(args[i])) && (false ==
swcmdflg))
                {
                    switch (args[i].ToLower())
                    {
                        case "-zero":
                            retType = retcodeType.allways;
                            break;

                        case "-any":
                            retType = retcodeType.anysafe;
                            break;

                        case "-all":
                            retType = retcodeType.allsafe;
                            break;

                        case "-f":
                            if ((i + 1) < args.Length)
                            {
                                i++;
                                confFile = args[i];
                            }
                            break;

                        case "-wait":
                        case "-limit":
                            if ((i + 1) < args.Length)
                            {
                                i++;
                                try
                                {
                                    waitTime = 1000 *
int.Parse(args[i]);
                                }
```

```
                                catch (Exception)
                                {
                                    waitTime = 0;
                                }
                            }
                            break;
                        }
                    }
                    else
                    {
                        swcmdflg = true;
                        if (dummyfile != args[i])
                        {
                            ag.Add(args[i]);
                        }
                    }
                }

                cmds = ag.ToArray(typeof(String)) as String[];
            }
        }
}
```

Windows, batch, tool