Universal Shell Programming as a glue language and technology

ISSN 2187-9664

USP MAGAZINE

Vol.2
400Yen

よりぬき版

for the sophisticated shell scripters

第１特集
シェルスクリプト本部長の分散処理
Netcatで世界を股にかける

第２特集
USP友の会プログラミング道場
シェルスクリプトで
思考のストレッチ

Linux女子部、語る!
Agile2011現地レポート

TechLION誌上中継
最後のRubyKaigiの裏側を高橋征義が語る
RubyとRubyKaigiから学んだこと

From Editor

コミュニティの担う役割に注目が集まっています。
ルールに縛られて、管理された下で行動するのではなく、
個々の意志によって、人と人とがつながることで、
様々な成果が生まれています。

人と人が何かの仕事によって結びつく根底にあるもの。
それは「志」であると、私たちは思います。
今月取り上げたコミュニティも、それぞれ何かの志を持って、
広く世に何かを訴えかけようとしています。

言語開発者の支援、女性エンジニアの地位向上、Agile の普及…。
これほどまでに、コミュニティ活動が活発な業界が他にあるでしょうか？
さまざまな志が、IT 業界のいたるところで芽吹こうとしています。

志の芽を見つけ、大切に育てていくこと。
これが、雑誌の役割の一つなのでしょう。

そして、私たち USP MAGAZINE も、
以下の 5 つの志を掲げ、この変動の時代をいかに技術者として生きていくべきかを、
皆さんと共に考える場でありたいと思っています。

（1）UNIX 哲学を学び、実践すること（Simple is Beautiful ）
（2）自律した技術者による協働を目指すこと（Independent Engineers）
（3）手しごとと工夫を大切にすること（Craftmanship）
（4）自然であること、Free であること（OpenMind）
（5）本質を見極め、自分の頭で考え行動する技術者たること（Sophisticated Engineers）

USP MAGAZINE 編集部

Contents

※ 薄字で記載している記事は、よりぬき版には収録されておりません。

特集1

Netcatで世界を股にかける

シェルスクリプト本部長の分散処理

執筆　リッチ・ミカン（松浦リッチ研究所）
技術協力　上田隆一（ユニバーサル・シェル・プログラミング研究所）

前号のシェルスクリプト部長の活躍は見てもらえただろうか。
彼は、マルチコア化が進んで多数の部下（＝論理 CPU）を使えるようになった背景に着目し、卓越した能力と機転の利いたアイデアで、膨大な量のミッションを首尾よくこなした。だが、彼の実力はその程度ではなかった。成果が認められて本部長に昇進した彼は、ネットワークが普及した現状を捉え、世界中の支社（＝ホストコンピューター）の人材を統括し、更なる成果をもたらすのだった。
というわけで今回は、シェルスクリプトによる分散処理を紹介する。
特殊なツールに頼らず、アイデア一つで勝負するのが部長の持ち味なのだが、分散処理においてもそれは変わらない。

君は本当に何でもできる奴だな

シェルスクリプト部長シリーズ、おかげさまで前編を読まれた方から「こんな手軽に並列化ができるのか！」と一部で好評を頂くことができた。名前付きパイプと “&”（バックグラウンド実行）というありふれた機能を使うという、言われてみれば何てことの無い手法だったからだろう。しかしそのように、手軽で、何てことの無い手法であることが重要なのだ。そうであるからこそ定番レシピ、つまり常套手段として普及できるのだから。

レシピといえば、ちょっと宣伝になるが、先月「シェルスクリプトシンプルレシピ 54」という書籍を USP 出版から発行させていただいた。さすがに本誌で紹介しているような並列処理とか分散処理など高度な活用法はカバーしていないが、数字や文字の加工法、行や列ごとの処理方法、ファイルの操作方法などを網羅している。他言語では言語仕様あるいは組み込み関数等で当たり前に用意されている基本処理を中心に逆引きリファレンスにした本だ。「シェルスクリプトは MS-DOS のバッチファイルに毛の生えた程度のもの」と開発言語として見てこなかったとしたら、その考えががらっと変るかもしれない。

話を戻すが、シェルスクリプトは何でもできて実に頼もしい奴だ。驚くなかれ、なんと音声合成して楽器の演奏もできれば、点や線を引いて図形も描けるし、Excel や PDF をも合成できるのだ！もちろんシェル自体にそのような機能があるわけではなく、各機能を呼び出しているに過ぎないが、シェルと親和性の高いコマンドが多数開発されている。これらの話は今後、本誌でも少しずつ紹介してきたい。

作図もできて……

Excel も使える。

何でもできて頼れるボス。

いや、それ誰でも大抵できるから

ではそんな、頼もしいシェルスクリプト部長の後編「本部長編」をお届けしよう。物語は本部長に昇進した当日から始まる。

本部長、新たなミッションを命ぜられる。

本部長への昇進は単なるご褒美ではない。更なる高度なミッションを任せるためだ。就任当日、シェルスクリプト本部長は息付く暇も無く新たなミッションを命ぜられた。急激に伸びる自社サービスへの需要に応えるため、各国に存在する支社のリソースを動員して対応能力を上げよというミッションだった。

ここで一旦、本部長の物語から現実的な話に落とし込んでしまって申し訳ないが、前編のおさらいを兼ねて上記のミッションが意味している実際の話を説明しておく。上記でいうサービスとは、前編でも扱った 1 億行もの膨大なレコード数のデータ（→**リスト 1**）を加工する作業のことであり、また各国に存在する支社のリソースとはネットワークに繋がった何台ものホストコンピューターのことである。

リスト 1．前編に引き続き扱う、１億行データ（抜粋）
番号 , 都道府県名 , 金額 , 年月日の4フィールド構成

```
1534 高知県 -9,987,759 2001年1月5日
3772 鹿児島県 5,689,492 1992年5月6日
8238 大分県 1,099,824 2010年2月22日
1292 秋田県 -4,497,866 1991年2月23日
9812 新潟県 5,721,468 2001年2月28日
6783 大分県 6,120,180 1992年2月14日
9779 高知県 -8,281,463 2005年5月15日
4623 千葉県 9,122,213 2010年3月15日
2383 兵庫県 -2,773,561 2007年3月28日
1306 和歌山県 -988,312 2008年12月7日
8423 島根県 -397,852 2006年11月3日
4054 沖縄県 9,754,147 2009年6月27日
0875 青森県 3,480,052 2000年6月4日
9864 石川県 -5,821,409 1993年5月31日
:
```

つまり、前編で取り扱ったデータを、今度はさらに複数台のコンピューターで分散処理させて、一層の処理速度向上を図ろうという話だ。

今回はコンピューター４台で挑む（→写真１）。各コンピューターのスペックは Intel Xeon W5580（3.2GHz 4 コア）を 2 基搭載で 8CPU 相当、メモリ 48GB と、前回と同様である。同性能のコンピューターが 4 台に増えるのだから、単純に考えれば処理速度は 4 倍である。前回、１台で実験した際、4 時間 46 分かかっていたソート処理を、8 プロセスによる並列化で 1 時 間 16 分へと実質 3.7 倍のスピードにすることができたが、今回はさらにそこからどれだけ早くなるだろうか。

写真１.
本部長が活躍するコンピューターの近景。本当はあと2台あるのだがちょっと離れた場所にあって撮れてない。だが、見た目は同じ！

SSH で通達、Netcat で搬送。汎用品で事足りる。

並列・分散処理を考える時は「人間のグループ作業だったらどうするか」を考えるところから始めるとやりやすいということは前編で示したとおりだ。人間がやる場合、今回の話であれば次の 2 つの作業を思いつくだろう。一つは支社に作業内容の通達を出すこと、もう一つは本社ー支社間における品物（データ）搬送経路を用意することだ。当然本部長もこの 2 つは考えた。だが、こんなことは誰でも考える事だ。

既存品を用いて導入コストを抑えよ

シェルスクリプト本部長が見せる真のアイデアはその先にあった。作業通達にしろデータ搬送にしろ、目的に応じたソフトウェアを開発すればできるに決まっている。だが本部長は、既存のコマンド群で実現できることに気づき、導入コストをかけずにこの 2 つを実現する方法を提案したのだった。

本部長のアイデアはこうだ（→図１）。まずは SSH を使う。SSH といえばリモートログインでよく使うコマンドだが、リモートコマンド実行やポートフォワーディング、最近ではレイヤー 2 VPN にまで対応する等、実に様々な機能を有している。今回は、このうちリモートコマンド実行の機能を利用する。これを使って支社コンピューター上で並列処理させるコマンドを実行させればよい。これで作業通達ができてしまうわけだ。ではデータ搬送はどうするか。既に名前を出しているが Netcat を使えばよい。Netcat はその名のとおり、いわばネットワークに対応した cat コマンドであり、標準入出力を TCP や UDP ポートに繋ぐことができる。従って、Netcat にポートリッスンさせて本社からデータを受け取り、それを標準入出力経由で加工用コマンドに渡し、処理後のデータを再び標準入出力経由で Netcat に渡して本社に送り返すのだ。これまた聞けば、何てことの無い手法ではないか！

Netcat（SSH も）は TCP/IP の上に実装されるコマンドである。つまり世界中に張り巡らされたインターネットを、そしてその先に接続された世界中のコンピューターを行き来させることができる。まさに、Netcat で世界を股に掛けるというわけだ！ *1

*1 Netcat は流すデータを暗号化しないので、実際に世界を股に掛ける場合は、流そうとしているデータのセキュリティには十分注意すること！

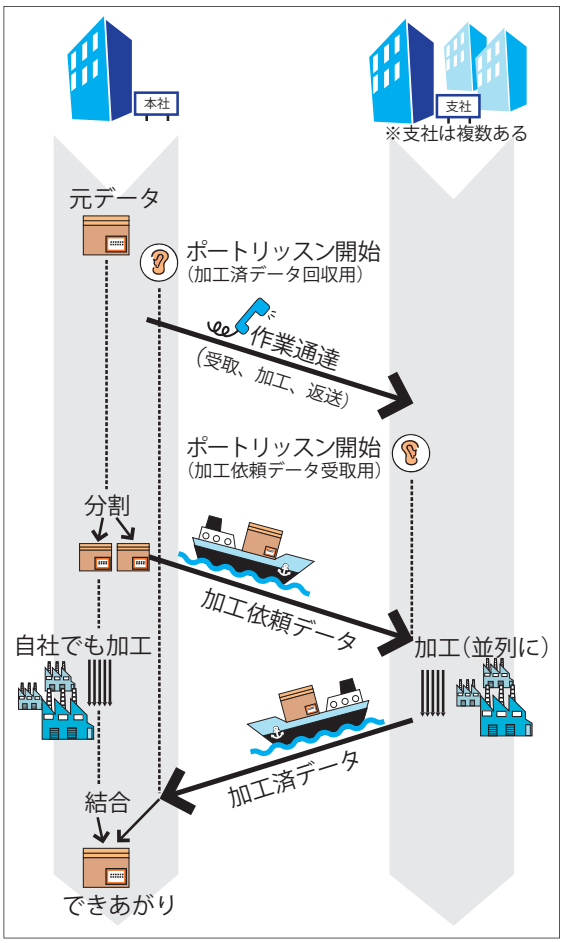

図１. 本部長のアイデア

本部長が出したプラン

このアイデアを利用し、本部長が提案したシェルスクリプトがリスト2[*2]である。既にコメントとしてリスト内に説明がなされているのでポイントのみ掻い摘んで説明する。

作業通達と搬送経路の確保という2つの仕事が必要であるが、実際に行う場合、先にやらなければならないのは後者である。ネットワークでA →→ Bのうような経路を構築する場合、終端である"→B"を先に作らなければならない。始端である"A →"は終端が既にできていることをと期待しているからである。今回のSSHによる作業通達(26～28行目)の中には始端を作る作業が含まれている。よって、加工済データ回収のためのリッスン用Netcatを、2)のSSHよりも先に1)で実行する。また、それらは"&"を使ってバックグラウンド実行させるようにしているのもポイントだ。そうしなければリッスンが終わるまで先へ進めず、リッスンを終わらせるためのデータが流せないので沈黙してしまう。

それが済んだら3)以降で、加工依頼データの搬出と回収行うわけだが、3)では自分を含めた4台のコンピューターで均等に分担させられるように仕分けをしている。ここではソート対象のフィールド値の大小で仕分けている(39～42行目)が、値が均等に分布しているならウマい仕分け方といえる。さすがアイデアマンだ。

ただし、このプランには少々横着している点がある(誌面の都合でというのもあるが)。一つはsleepコマンドを使っている箇所(14,29行目)だ。前者は本社コンピューターのリッスンがちゃんと開始するであろうことを期待した待ちであり、後者は支社コンピューターがちゃんとリッスンを開始するであろうことを期待した待ちである。逆に言え

*2 リスト2の28,50行目に"> /dev/tcp/x/y"という記述がある。これは"| nc x y"に相当するLinuxカーネルの機能だ。カーネルレベルの実装の方が速いのでこの記述を採用した。

リスト2. 4台のコンピューターでソートを分散処理するスクリプト"distsort.sh"

```
#!/bin/sh

### 1) [ 加工済データ回収に備えた準備 ] ###########
#
#  部下から計算結果を受け取るため、
#  Netcat でポートをリッスン状態にしておく。
#
for i in 1 2 3; do
  # --- 支社 n からは ポート 10000+n で受け取る ---
  nc -l 1000${i} > ./DataFrom${i} &
done
sleep 1 # きちんとリッスン状態にさせるため小休止

### 2) [ 業務の通達 ] ############################
#
#  SSH を使い、各支社に、以下の業務を通達する。
#
#    1. 本社からの加工依頼データを受け取るため
#       Netcat でポート 10000 をリッスンせよ
#    2. 届いたデータはパイプ経由で parasort に送り
#       並列ソートせよ
#    3. 加工したら、各支社用に本社でそれぞれ
#       リッスンしているポートに送りつけよ
#
for i in 1 2 3; do      ↓ 26～28 行目：作業内容
  ssh shisha${i} "nc -l 10000                | ¥
                  /home/UTL/parasort -k1,1   ¥
                  > /dev/tcp/honsha/1000${i}"
done
sleep 1 # 通達が行き渡るであろうまでの間小休止

### 3) [ 加工依頼データの仕分け ] ################
#
#  各支社に、送る加工依頼データを仕分けする。
#  ( 元データの、第一レコードの値の大きさに応じて
#    仕分けしている )
#
cat ./TESTDATA          ↓ 39～42 行目：仕分け | ¥
awk '{if     ($1<"2500"){print $0 > "./data0"}  ¥
      else if($1<"5000"){print $0 > "./data1"}  ¥
      else if($1<"7500"){print $0 > "./data2"}  ¥
      else              {print $0 > "./data3"}}'

### 4) [ 加工依頼データの搬出 ] ##################
#
#  各支社に向け、加工依頼データを搬出する。
#  同時に、本社でも加工をする。
#
for i in 1 2 3; do
  cat ./data${i} > /dev/tcp/shisha${i}/10000 &
done
cat ./data0 | parasort -k1,1

### 5) [ 回収した加工済データの結合 ] ############
#
#  各支社からの加工済データの到着を待って、
#  結合 ( 続けて標準出力に送出 ) をする。
#
for i in 1 2 3; do
  pid=`ps axww | grep "nc -l 1000${i}" | ¥
       grep -v grep | awk '{print $1}'   `
  if [ -n "$pid" ]; then
    wait $pid 2>/dev/null
  fi
  cat ./DataFrom${i}
done

### 6) [ 後始末 ] ################################
#
#  作った一時ファイルを全て削除する。
#
rm -f ./DataFrom[1-3]
rm -f ./data[0-3]
```

表1．parasort 及び 4 台分散 parasort の処理速度と、前者に対する後者の実質的な速度向上

処理行数	1 億	1000 万	100 万	10 万	1 万	1000
parasort（前回）	4590. 105 秒	349. 058 秒	19. 189 秒	1. 044 秒	0. 334 秒	0. 317 秒
上記 +4 台分散	1243. 445 秒	102. 598 秒	9. 530 秒	3. 536 秒	3. 332 秒	3. 297 秒
速度向上	3.7 倍	3.4 倍	2.0 倍	0.30 倍	0.10 倍	0.10 倍
（参考）単独 sort からの速度向上	13.8 倍	10.4 倍	4.6 倍	0.46 倍	0.013 倍	0.001 倍

※ 各コマンド完了までの秒数を time コマンドで 5 回計って平均したもの。分散しない parasort 測定値は前編の表から引用。尚、行数の調整には head コマンドを使用。

ば、本当に開始したかどうかを確認せず、「これくらい待っておけば大丈夫だろう」と見切り発車をしてしまっているのである。コンピューターの台数が多かったり、相手がインターネット経由の遠いところにあるのであればちゃんと確認をとるコードを付け加えておかねばならない。

もう一つは、前にこのスクリプトを実行して途中で中断させてしまった場合への備えである。この時、本社や支社でポートリッスンがされたままの状態になっている可能性があるので、それを確実に終わらせておかねばならない。

これらに気をつけないとデータに欠損が起こったりスクリプトが沈黙したりする。実用化するなら、最低限その対策を施さなければならない。

そして本部長は、結果を出した。

さてこのシェルスクリプトの効果はどれほどなのであろうか。今回は 4 台の同性能なコンピューターを使って分散処理を行ったので、理論的には 4 倍強の処理能力となる。果たしたどれだけこの数字に迫ることができるのであろうか。前回同様にベンチマークを実施した（→**表1**）。

前編と測定方法を揃えて比較できるようにした。すなわち、1 億行〜 1000 行までそれぞれ 5 回計測した平均をとった。ただし、前編では AWK と sort の 2 種目で実施したが、誌面の都合もあって今回は sort のみとした。

1 億行時の処理時間は、およそ 1243 秒（20 分 43 秒）となって実質的な速度向上は 3.7 倍を達成した。前述のとおり理論的な能力向上が 4 倍強であるから、分散化によるオーバーヘッドは少なく、かなりいい結果を出したと言える[*3]。こうしてまたシェルスクリプト部長改め本部長は、命ぜられたミッションに対して申し分の無い結果を出したのだった。

*3 もちろん行数が少ない場合はこの限りではない。効果が逆転する分岐点が 100 万行と 10 万行の間に存在しており、並列化時よりも大掛かりな様が伺える。

エクストリーム‼ 26台のマシンで処理に挑んだ。

今回の計測とは別なのだが、シェルスクリプト本部長はその後、**図2**に載せたなんと 26 台コンピューター群を舞台にしたとてもエクストリームな分散処理プロジェクトを任されることになった。仕事のデキる人物には次から次へと大きな仕事が舞い降りてくるものだ。

前編・後編と二回に渡ったシェルスクリプト部長シリー

図2．26 台ものコンピューターから構成されたエクストリームなマシン

ズの最後に、少しだけこの話[*4]を紹介したいと思う。

分散処理にも課題がある

1 台のコンピューターの論理 CPU 数の増加を期待した並列処理の場合だと、いくら資金があっても CPU 数を増やすアプローチはすぐに限界に達してしまう。一方分散処理なら予算に応じていくらでも台数を増やせる。まさに分散処理だからこそなせる技だが、その分散処理にも、単純に台数を増やしていくだけではやがてぶつかる壁がある。それは加工依頼データの搬出と回収だ。

例えば分散させる先のコンピューターが n 台あったとしよう。そして分散元のコンピューターと分散先の n 台のコンピューターがハブ等を介して一つの LAN に接続される形になっていたとする。この状態で分散処理させたデータを n 台のコンピューターに一斉に送受信する様子を想像してみてもらいたい。ネットワークのスループットは分散元の親機の口で n 分の 1 に落ちてしまうことが容易に想像できるだろう。だが、今回測定した時のように分散先の台数 n が少なければ問題にならない。なぜかといえば、CPU で加工処理する速度、つまりは加工処理のスループットがネットワークのスループットに比べて無視できるほど低いからだ。実際に表 1 の測定結果を見てみると、分散処理化する前の 1 億行ソートでは 1 時間 16 分かかっている。1 億行のデータは約 4GB あるので、加工処理時のスループットは約 7Mbps となる。これは Gigabit Ether の帯域を 3 台で分割した場合のスループットに比べて 2 桁も低いことがわかる。

しかし分散処理に協力してくれるコンピューターが増えればネットワークのスループットの低下がしていき、やがて加工処理のスループットを下回ってしまう。今回の場合、分散先は 25 台ということになるので、Gigabit Ether だと 40Mbps にまで落ちてしまい、仮に 1 台あたりの性能が同じ 7Mbps だとしたら、もう無視できない値だろう。

課題の克服も、やはりアイデア

では、この 26 台コンピューター群ではそれをどう克服したのか？答えはネットワークの直列接続だ。しかも InfiniBand という超高速なインターフェースを 1 台につき 2 個用いたという。具体的なアイデアは**図 3** に示した。

搬出時は、受信したものを tee コマンドに掛け、自機のディスクに落とすと共に次のホストに再搬出する。一方回収時は、受信したものを cat コマンドで受け、自機の加工済みデータを追加して次のホストに送る。こうすればスループットは落ちない。レイテンシー（遅延）は増えるもののスループット低下に比べれば殆ど問題にならないという。

肝心の結果はというと、なんと 1 億行ソートをわずか 2 分で完了したというのである。コンピューターのスペックが違う（詳細は聞けなかった）ので単純に比較はできないものの、分散処理も並列処理もなしの時には 4 時間以上かかっていたことを鑑みると驚異的な早さだ。これは確かにハードウェアに資金を投入したことによる効果も大きいが、数々のアイデアが揃わなければ実現できなかったものだ。

ハードウェアもシェルスクリプトも、アイデアによって活かされる。シェルスクリプト本部長はいつも周囲にそう語るのだった。

*4 このエピソードは独立行政法人新エネルギー・産業技術総合開発機構（NEDO）による平成 21 年第 2 回イノベーション推進事業「研究開発型ベンチャー技術開発助成事業」の助成対象事業の、「パイプライン計算機とユニケージによる高速情報処理技術の実用化に関する研究」の一部として実施された研究活動に基づくものである。

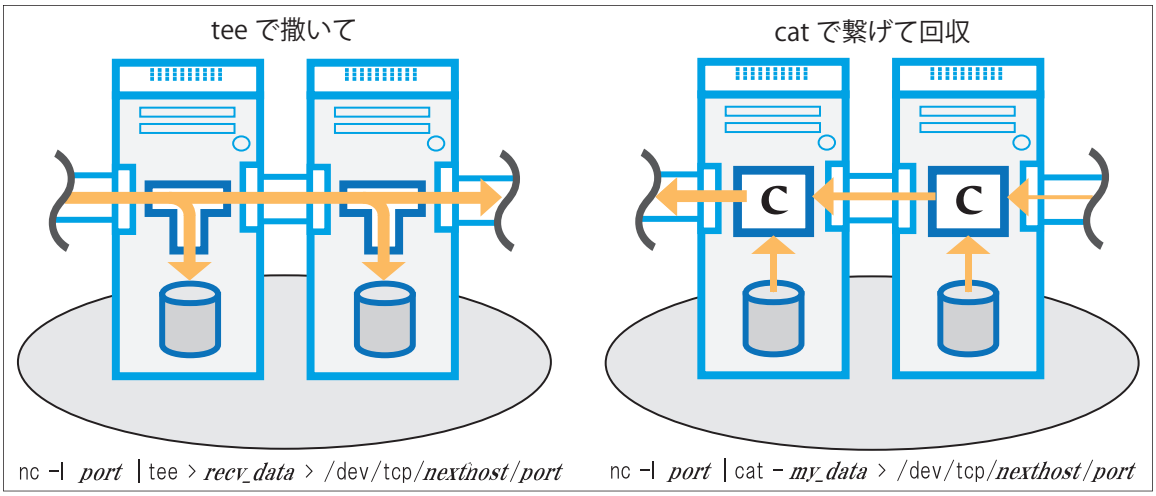

図 3．分散処理コンピューターの台数が多い場合は、直列なネットワークにする方が有利

シェルスクリプト大喜利 第二回

司会：『吹く』編集長・みかん

さて次は、３ヵ月間のご無沙汰、シェルスクリプト大喜利（略して sh 大喜利）のコーナーでございます。楽しみにしてくださっていた皆様、そして作品を投稿してくださっていた皆様、お待たせ致しました。まずはメンバーのご挨拶から。

私は司会を務めます吹く編集長のみかんです。誤字ではありませんよ。思わず吹いてしまうような作品を募集し始めて約半年、このコーナーも無事二回目を迎えることができました。ありがとうございます。ついに二桁突入です（二進数で）。次は三桁目指します！（目標低っ）

他のメンバーは「読者の皆さん」、というわけで読者参加型のコーナーなのです。ではシステムを説明します。まだ連載回数も少なくルールが定着していませんので、同じ内容ですが暫くは掲載し続けます。

シェルスクリプト大喜利とは

購読している皆さんに毎号お題を出して回答を募り、面白いものを紹介します。そして、良い回答には段位を授与、悪い回答では段位を強制返還させてもらいます。面白さに応じて授与・返還される段数は様々です。そして、見事十段になりますと‼ **筆舌に尽くしがたいちんじゅうちゃんグッズ**をプレゼント！日曜夕方のアレとほとんど同じですね。ただ、扱うものがシェルスクリプトということでいくつか特有のルールがあります。以下にそれを記します。

シェルスクリプト大喜利特有のルール

一、sh 大喜利はクイズやテストではありません。つまり決まった答というものはありません。敢えて言うなら面白いスクリプトが正解です。

二、面白いスクリプトとは例えば、こんなものです。

イ、人が考えつかない意外性がある

ロ、美しい or 芸術的 or 記述がシンプル・短い or 高速

ハ、アイデア・こだわりが光る

ニ、ネタになるようなバカバカしさ、くだらなさがある

などなど、ただし最後のは段位強制返還の恐れが…。:-)

三、スクリプトの動作環境は Linux とします。そして、とくに断りがない場合は、Linux JM(http://linuxjm.sourceforge.jp/) に記載されているコマンド及び機能のみ使用可能とします。これは多くの人が楽しめるようにするためです。(JM にあるという理由で、**C シェル系での回答も OK！**)

四、sh 大喜利はシェルスクリプトを披露する場ですから、**Perl や Ruby、Python などは使っちゃダメ**です。そもそも JM にも載っていません。逆にシェルスクリプトにとって不可欠な awk や sed 等は OK です。JM にもありますし。

それでは、本日のお題に進みましょう。

本番開始

<第一問>

strings コマンドの意外な使い方を教えてください。

前回に引き続き、マイナーなコマンドに光を当てて、使い道を考えてあげようというお題です。

今回光を当てるのは strings コマンド。コンパイル済みのバイナリファイルはあれどソースファイルは無いという時、そのバイナリファイルの中から元のソースファイルにあったであろう文字列を探し出す、というような使い方をされがちなコマンドです。これだとソースを無くしてしまった（or 引き継ぎが出来なかった）開発者が涙目でデバッグする時くらいしか使い道がありません。何かもっといい活用法はないものでしょうか。

といって募集しましたが、回答は辛うじてお二方だけでした。ちょっと難しかったですかね……。まずはお一人目

◎ゆきとさんの回答

```
#! /bin/sh
LANG=C
NGword='buggy'
cmds=`grep -r $NGword $(echo "$PATH" | tr : ' ') 2>
/dev/null | strings | awk -F '[: ]' '/^Bin/{print $
3} /^¥//{print $1}' | sort | uniq`
for cmd in `echo "$cmds"`; do
  echo "=== $cmd ===";
  strings $cmd | grep $NGword; echo
done
```

添えられていた説明によると、パスが通っているディレクトリにある全てのコマンドのうち、"buggy" という記述が埋め込まれているアヤしいコマンド、および問題箇所を列挙するとのこと。それから、単純に一つ一つ strings を実行すると遅かったので検索の早い grep で一旦舐めて、引っ掛かったものだけ strings したという。

うーん、「意外な用途」かというと、それほど意外ではないかなあ。だけど、実行結果を見ると「RS/6000 版 AIX 3.2 上の sh はバグってる」とか「Solaris 8 の bash 2.03 はバグってる」とかいろいろタレ込みが埋め込まれてて面白い。このお題への投稿数も少なかったし、よし、**初段授与**！

◇ ◇ ◇

もう一人の回答は？それがなんと、第二問と一緒に解いたというのですよ。そう来ましたか！なので、次のお題へ進みます。

<第二問>

vol.1 の 6 ページに出てきた interlace コマンドをシェルスクリプトに移植してください。8 つのストリームを重畳する専用バージョンで構いません。

interlace コマンドとは、8 つのファイルの一行ずつ順番にマージして 1 つにするために作られたオリジナルコマンド。これは難しいだろうと思い「段位はサービスしますよ」とか言って投稿募集したんですが、出題者である私の予想を裏切って簡単に解く回答の投稿が続出！参りました。段位サービスさせていただきます……。

◎@hirochachacha さんの回答

```
paste -d ´¥n´ a.txt b.txt c.txt d.txt e.txt f.txt g.txt h.txt
```

コメント「ワンライナーで十分です」。はい、読者の皆様御見それしました。そーなのですよ！ paste コマンドを使えばいいだけの話だったのです。うぇ〜い、みんな段位もってけー。というわけで、**三段授与**させていただきます。

◎齊藤明紀さんの回答

```
#!/bin/sh
paste -d´¥n´ "$@"
```

```
#!/bin/sh
(
while read a ;do
echo "$a"
read a <&3 && echo "$a"
read a <&4 && echo "$a"
read a <&5 && echo "$a"
read a <&6 && echo "$a"
read a <&7 && echo "$a"
read a <&8 && echo "$a"
read a <&9 && echo "$a"
done
) < "$1" 3< "$2" 4< "$3" 5< "$4" 6< "$5" 7< "$6" 8< "$7" 9< "$8"
```

最初の回答が paste コマンドを使ったシンプルな書き方、後の回答が paste コマンドを使わなかった場合の回答だそうです。

そう！私は後者のような回答がくることを予想していたのですよ。ファイルディスプリタ番号は 0 ~9 までありますが、使えるのが 1 と 3 〜 9 で丁度 8 個なのです。出題者の意図まで読まれていたとはもう御見それしまくりだ。もちろん前者の回答も素晴らしい。というわけで、えーい**四段授与!!**

◎hamano さんの回答

```
#!/bin/sh
paste -d´¥b´ "$@" | strings -1
```

コメント「1 問目と 2 問目を同時に解きました」。ということで、これがその複数のお題を同時に解くという回答です。複数のお題を一遍に解くという芸当は、まるでお笑いマ○ガ道場の富永○朗みたいじゃないか。strings の使い方は正直「そのまま素通ししてるだけじゃん！」と思うわけだが、それゆえにコロンブスの卵的意外性にやられたし、同時解きという技を披露されたし、もちろん paste コマンドを使ってシンプルに解いてるし、これも私の予想をいくつも裏切っていてすばらしい。**四段授与!!**

◇ ◇ ◇

他にも、paste をコマンドを使わずに至極真っ当に解いてくださった回答も多数寄せられましたが、paste を使うという解法に圧倒されてしまい、採用に至りませんでした。ごめんなさい！では最後のお題。

<第三問>

USP 友の会に会員登録すると事務局に次のメールが自動送信されるそうです。

```
タイトル:USP友の会ご入会ありがとうございます。
本文:
XXXXXXXX
---以下、事務手続き用          (注)
name=山田太郎                ← この4項目の順番は
email=xxx@usptomonokai.jp    ← 仕様では未定義で、
place=東京都                 ← どの順番でくるか
how=twitter                  ← わかりません。
---
```

これを name, email, place, how の順番で /home/usp/members.csv に CSV で追記出力する一行野郎スクリプトを作ってください。（コードはなるべく短く!）

一行野郎シリーズの第二問です。レベルとしてはまだまだ簡単でしょうか。もともと半ば解決を依頼される形で問題提案されたのですが、簡単だったため一行野郎問題にしてかつ、橙色のような制約も追加してみました。これにより、単純に行数だけを見て値を扱おうとすることができないようになっています。

◎さとうふみやす（@satoh_fumiyasu）さんの回答

```
tail -n5 |sort -r |(while read -r l;do set -- "$@"
"${l#*=}"; done; echo "$2,$4,$1,$3" >>/home/usp/mem
bers.csv)
```

kshでもzshでも）版も投稿してくださいましたが、Bash版の方が文字数が若干少ないこと、またJMに載っているコマンドというこのsh大喜利のルールにより、Bash版を採用しました。

ふむふむ、これに届いたメールの本文をパイプで流し込めば書き込むというわけか。そしてソートを掛けることで4つの変数がどの順番で来ても正しく並ぶようにしている。お題ではなるべく短くということなのでスペースを削って欲しかったところだが、それ以外はまさにお手本どおり、**初段授与**！

◎@hirochachacha さんの回答

```
cat ~/mbox|(while read l;do eval $l;done;echo "$nam
e,$email,$place,$how")>>~/members.csv
```

これはちょっと面白いです。会員データのある行は丁度代入式になっているので、evalを使ってうまいこと利用しています。こうすればsortは要らないというわです。

実は代入式をそのまま利用するというのはちょっと期待していたんだ。それに、ファイルパスがチルダ表記になっているところを差し引いても先の回答より文字数が少なくて、より望ましい回答だということで、第二問の回答分に二段加えて**五段を授与！**

そして最後にオシい！という回答を紹介。

◎gori.sh さんの回答

```
echo `grep =|tr nh []|sort|sed s/.*=//|tr []¥¥n nh¥
`>/home/usp/members.csv
```

これは恐らくお題を読み間違えてしまったのだと思います。間違いが2つあり、一つはカンマ区切りにしていないこと、もう一つはファイルへの書き込みが追記ではなく新規になっていることです。ただ、始めから字数を節約することを意識してくれていまして、巧妙なことをしていて短いのです。指摘されたその間違いを解消するためのコードを書き足しても恐らく一番短かったと思います。

お題の要求を満たしていないので残念ながら段位進呈はなし。オシい！

◇　◇　◇

というところで本日の大喜利はこれにてお開き！読者の皆様、投稿してくれた全ての皆様、ありがとうございました。

投稿大募集!!

シェルスクリプト大喜利は皆様からの投稿あってこそ成り立つ企画です。お題に対する回答、お題そのものを広く募集致します。さて、次号のお題はこちらです。

次回のお題

一、引数として与えられた平仮名（全角UTF-8）が回文であるならば戻り値（$?のこと）として0を、そうでなければそれ以外の値を返すシェルスクリプトを作ってください。尚、環境変数としてLANG=ja_JP.UTF-8がセットされているものとし、これに対応しているコマンドを使って構いません。また、日本語における回文のルール「濁音、半濁音、促音、拗音は清音と同一視」に従って判定してください。コードが短いとか美しいとか、何かしら見応えのあるものがいいです。

二、シェル変数$Tには現在の温度（小数点第1位まで）が、同じく$Hには現在の湿度（整数）が入っているものとします。この2つを元に不快指数を計算し、

温度 =nn℃ 湿度 =nn% 不快指数 =nn

というフォーマット（温度と不快指数は小数点以下四捨五入して表示、また行末は改行すること）で表示する一行野郎スクリプトを作ってください。なるべく短い方がいいです。尚、不快指数の計算式は

$0.81T+0.01H(0.99T-14.3)+46.3$

とします。

三、ちょっと志向を変えて、本当の大喜利っぽいお題を出したいと思います。

「『こんなシェルスクリプトはカッコいい』一体、どんなスクリプト?」

コードの見た目がカッコいいのもよし、実行結果がカッコいいのもよし。

投稿のしかた

お題への回答は、お名前（ペンネーム）、回答したいお題番号、回答スクリプト、簡単な補足の四点セットで下記の宛先へ！一人何問でも何個でも回答可です。尚、次回締め切りは**11月28日（月）午前0時**とします。しかもその間は何度でも回答の修正を受け付けます。

お題もどしどし送ってくださーい

それからお題も大募集。考えてくれた方にも段位を授与します。自分で出題して回答するもの、今のところ可！

投稿先

どちらも投稿先は、mag@usptomonokai.jp です。

本誌をイベントで購入された方も、ぜひご参加ください。

USP MAGAZINE vol.2
編集・発行人　鹿野恵子　発行・発売 USP 友の会　〒105-0003 東京都港区西新橋 3-4-2 SS ビル 3 階
有限会社ユニバーサル・シェル・プログラミング研究所内　電話 03-3432-1174
定価 400 円（本体価格 381 円）