株式会社 ITS MORE
2020年4月設立

Digitally signed by 佐藤 豊
Date: 2020.08.01 12:58:10 +09'00'

2020年7月31日 投稿者: SATOXITS

全く新しいアーカイブフォーマット

社長：ただいまー！

経理：酒臭いですね。

社長：12LSUを投じたメンタルヘルスケアプロジェクトを完遂して参りました。

経理：ストレススーパーフリーなのに。

社長：あそれで、マスターから仕入れた話なんですが、あの新庄君が現役復帰とかなんとか。

基盤：新庄 現役で検索…　確かにそういう話はあるようですね。

開発：まだ48歳みたいだし、野球という競技ならあり得ない話では無いですね。

社長：明るい話題ですねー。私はネット記事で栗山くんのしょぼくれ様を見かけるたびに気分が暗くなるのです。

基盤：ﾉﾑさんも亡くなっちゃいましたしね。

開発：それで、ソースを検索するのに何十万のファイルをNFSでfind grepしてたらとんでもないなとは思うわけです。それに以前から、全文検索ってgrepで良くね？と思ってたところでもあり。ならばいっそ、検索用テキスト兼用のアーカイブフォーマットを作ればどうかと思ったところです。

社長：コンセプトは？

開発：単にgrepできる単一べたファイルです。複数ファイルを単一ファイルにするので、おまけにアーカイブとしての機能も持たせる。しかも単にcatしただけで結合できる。手

作業で編集しても壊れない。

基盤：shar とは違うんですか？

開発：shar だと、grep してもヒットした行がどのファイルだかはわからないわけです。

社長：それは以前言っていた、MIMEのマルチパートの話とは違うんですか？

開発：まーーったく違います。あれ？でもないかな？

基盤：さっきから思うんですが、アース渦巻って目に滲みますね…

開発：何にしても、改行コードで終わる「行」というのは、人間の認識と密接不可分にして、一種の可変長レコードとしてもリーズナブルなサイズである傾向があるわけです。

基盤：蚊取り線香は根っこの勾玉状のとこまでくると達成感を感じますね。あれまで全部キレイに燃えるのが技術の粋のような。

社長：時限爆弾甩にも優秀ですね。あれ？甩、用。なんでしょうねこのヒゲの生えた甩って。

開発：子供の頃は、導火線という単語に萌えましたね。

開発：ああ、それで、shar でもいいですよ別に。shellで実行すると分解される、要件はそれだけですよね。

開発：個別ファイル型的には、png のテキスト表現に萌えますけどね。

＊　＊　＊

基盤：ふぁぁぁああ。よく寝ました。

社長：もうちょっと寝を3回くらいやりましたねｗ

開発：それで寝ている間に思ったんですが、HTMLをアーカイブファイルとして使うのもあるかなと。

社長：HTML archive だから har 型式ですかね。

基盤：はあｗ

開発：これは以前検討した、ウェブページの魚拓型式のベースとしても有用だと思うんです。

開発：ただ、バイナリデータを入れるとするとやはり data uri かなと思うんですが、表現できるサイズの制約があります。

社長：メガバイト程度のHTMLデータなら今のブラウザならへっちゃらですけどね。

開発：span かなにか適当なタグにbase64くらいで入れてidつけてdispaly noneにして、JavaScript で取り出すっていうのが妥当な線かも知れません。

基盤：ファイル名だけで探したいことも多いですよね。ただ、特定のファイル型というか、拡張子だけでフィルタしたことも多い。

開発：となると、自分の中に埋め込まれたデータを検索するJavaScriptが必要だということですね。検索結果をテーブル表示したりとか。グラフ表示なんかもできると良いかな。

社長：検索結果の中で検索語にヒットしたところをハイライトして、その前後の文脈を表示するという機能は絶対必要ですね。

基盤：定義・参照関係がわかる文書なら、関数名のような単語がハイパーリンクになっているとうれしいです。

＊　＊　＊

社長：ああそれで、今週のあたまにやった片付けごとのたまが打ち返されてきてまして、要するにウェブサイトに配布ファイル群を置く場合に、HTTPサーバでディレクトリのインデックスを動的に作るのはまずいということらしいのです。サーバの運用上の問題かも知れません。

基盤：負荷の大きいサーバだと、性能面でも問題でしょうね。

開発：ああ、それなら find -ls の結果を HTML に変換するだけで済みますね。せっかくなので details タグを活用して。で、どのファイルですかね。

社長：月曜にやったこととか、もう完全に忘れてますからね…　ああ、Download にあるこの tgz かな？

基盤：というか、Desktop に原型があるようですが。

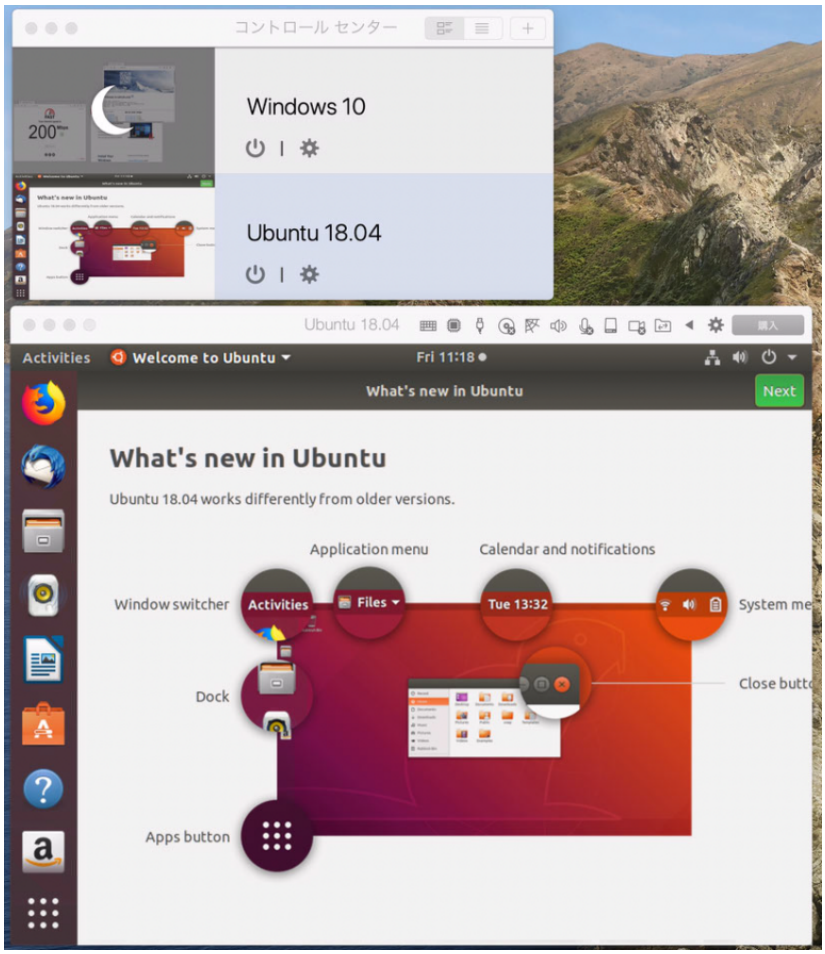
開発：ああ、これですね。

＊　＊　＊

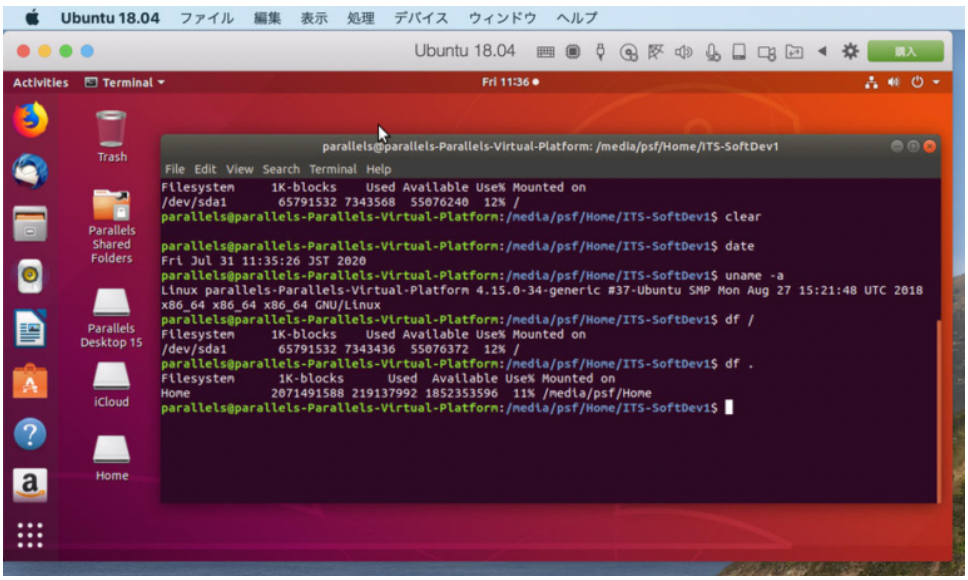
社長：ところで、Parallels は結構使えそうなので、あれに Linux を入れてブラウザの開発環境として使ってみてはどうかと思うのですが。

基盤：お試し版がまだ生きてるはず…　ああ、Ubuntu 18.04 がありますね。じゃこれをインストール。

基盤：ぐわー。なんですかこのお手軽高速インストール。ライトセールもびっくりしかもデスクトップ版。



開発：1分でダウンロード・インストールですね。確かに55MB/s 出てましたから、数GB程度のファイルなら出来る計算です。



基盤：なにかマウントされてるなと思ったら、ホストのiMacにマウントされてるドライブがデフォで見えるようになってるわけですね。デスクトップ画面が自在にリサイズするし。

社長：昔はVMwareでそうして使ってましたけどね。あと、ウィンドウ毎にホストの中に表示する、ユニなんとかいう機能とか。

基盤：うーん、このiMacにはSSDを2TB付けましたし、このまんまで Chromiumビルドできそうです。やっておきますか。Linux用のインストールの手引を再び開き…

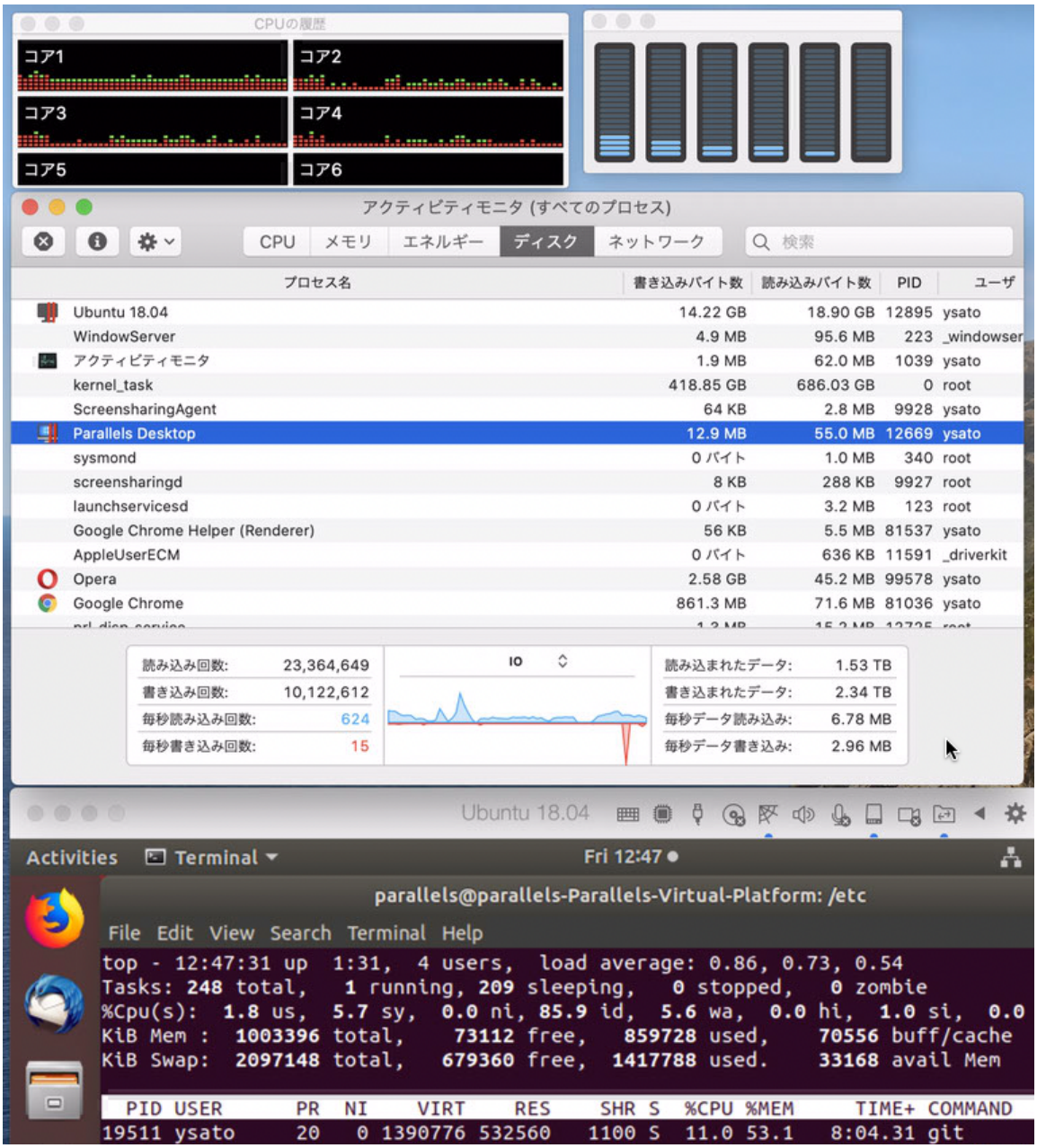
基盤：その前にVMの接続をブリッジにして…　sshログインできるようにして…　自分をsudoerに追加して…　git を apt install して…　では始めます。git で chromiumをクローン。速度の変動が大きいですが13MB/s程度ですね。

開発：一昨日のMac版では10分で終わった所、昨日のLinux版ではダウンロード後のdeltaの処理で2時間かかってしまった難所ですね。

社長：一昨日はLenovoがブーブー、昨日はQちゃんがゴロゴロいってましたが、今日のiMacは全く音なしですね。


```
pu18% date
Fri Jul 31 12:22:36 JST 2020
pu18% uptime
 12:22:43 up  1:07,  3 users,  load average: 0.05, 0.11, 0.09
pu18% time git clone https://github.com/otcshare/chromium-src.git
Cloning into 'chromium-src'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
Receiving objects: 100% (12999042/12999042), 14.30 GiB | 12.37 MiB/s, done.
remote: Total 12999042 (delta 2), reused 2 (delta 2), pack-reused 12999039
Resolving deltas:   0% (40368/10242849)
```

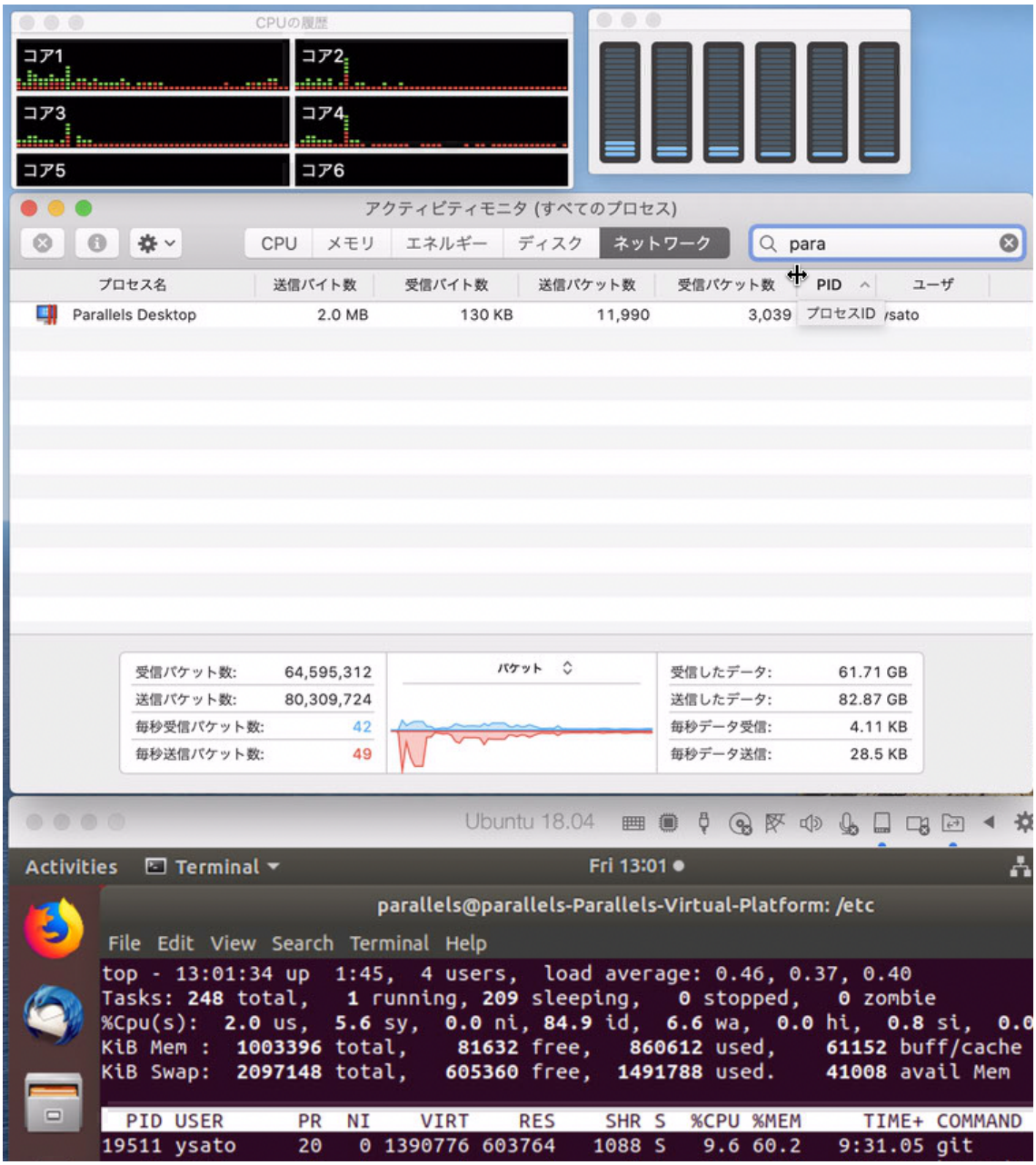
基盤：やはりclone自体は20分ですね。さて、これからデルタ地帯がどうなりますか。




```
pu18% time git clone https://github.com/otcshare/chromium-src.git
Cloning into 'chromium-src'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
Receiving objects: 100% (12999042/12999042), 14.30 GiB | 12.37 MiB/s, done.
remote: Total 12999042 (delta 2), reused 2 (delta 2), pack-reused 12999039
Resolving deltas:   2% (229893/10242849)
```

基盤：これは昨日と同じ、2時間半コースのペースですね。

開発：うーん、git もちびっとしかCPUを食ってないし、VMホスト直結のSSDを使っているので、ファイルアクセスもネックになってないと思うんですけどね…　ひょっとしてネットワークアクセス？



基盤：このガラ空き状態で時間がかかるって、ネットのレイテンシーかなって気はしますけどね。

開発：strace してみますか…　働いているのはさきっぽのプロセスだけですね。でこいつは何をやっているのかな…

```
pu18% ps xf
  PID TTY      STAT   TIME COMMAND
19456 ?        S      0:00 sshd: ysato@pts/1
19457 pts/1    Ss     0:00  \_ -bash
26885 pts/1    R+     0:00      \_ ps xf
15786 ?        S      0:00 sshd: ysato@pts/3
15787 pts/3    Ss     0:00  \_ -bash
19275 pts/3    S+     0:00      \_ git clone https://github.com/otcshare/chromium-src.git
19276 pts/3    S+     0:54         \_ /usr/lib/git-core/git-remote-https origin https://github.com/otcshare/chromium-
19287 pts/3    S+     0:22            \_ /usr/lib/git-core/git fetch-pack --stateless-rpc --stdin --lock-pack --thin
19511 pts/3    Sl+   10:02               \_ /usr/lib/git-core/git index-pack --stdin -v --fix-thin --keep=fetch-pac
14298 ?        S      0:00 sshd: ysato@pts/2
14299 pts/2    Ss+    0:00  \_ -bash
14224 ?        Ss     0:00 /lib/systemd/systemd --user
14225 ?        S      0:00  \_ (sd-pam)
pu18% sudo strace -t -p 19511
strace: Process 19511 attached
13:07:18 futex(0x7fe28c7959d0, FUTEX_WAIT, 22830, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
13:07:19 --- SIGALRM {si_signo=SIGALRM, si_code=SI_KERNEL} ---
13:07:19 rt_sigreturn({mask=[]})        = 202
13:07:19 futex(0x7fe28c7959d0, FUTEX_WAIT, 22830, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
13:07:20 --- SIGALRM {si_signo=SIGALRM, si_code=SI_KERNEL} ---
13:07:20 rt_sigreturn({mask=[]})        = 202
13:07:20 futex(0x7fe28c7959d0, FUTEX_WAIT, 22830, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
13:07:21 --- SIGALRM {si_signo=SIGALRM, si_code=SI_KERNEL} ---
13:07:21 rt_sigreturn({mask=[]})        = 202
13:07:21 futex(0x7fe28c7959d0, FUTEX_WAIT, 22830, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
13:07:22 --- SIGALRM {si_signo=SIGALRM, si_code=SI_KERNEL} ---
13:07:22 rt_sigreturn({mask=[]})        = 202
13:07:22 futex(0x7fe28c7959d0, FUTEX_WAIT, 22830, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
13:07:23 --- SIGALRM {si_signo=SIGALRM, si_code=SI_KERNEL} ---
13:07:23 rt_sigreturn({mask=[]})        = 202
```

基盤：おおっと意外。IOしてないみたいですね

開発：いや、でも端末へ1秒おきに進捗報告出しているんで、これのwriteがトレースに出てこないというのも不思議です。

```
19275 pts/3    S+     0:00      \_ git clone https://github.com/otcshar
19276 pts/3    S+     0:54         \_ /usr/lib/git-core/git-remote-htt
19287 pts/3    S+     0:22            \_ /usr/lib/git-core/git fetch-
19511 pts/3    Sl+   10:58               \_ /usr/lib/git-core/git in
```

基盤： git index-pack slow で検索…　Git はファイル数が多くなるとすごく遅いようですね。チェックサムに時間がかかっているとか？

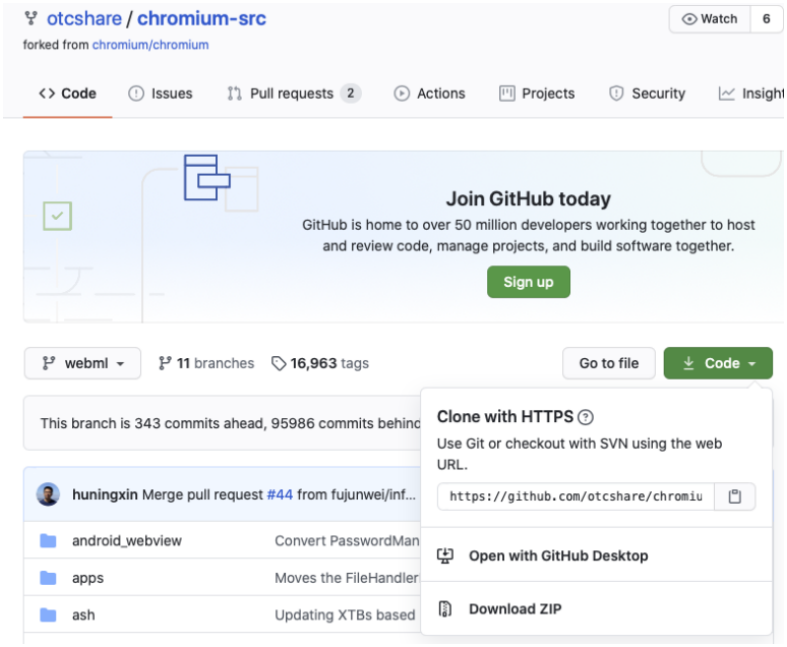
開発：いや、こっちはある時点のものをどんと持って来たいだけなんですけどね。20GB程度の。これ、kill しちゃって続けたら不味いですかね？まだ10%しか進んでない…

社長：kill。

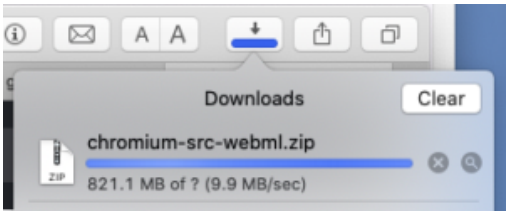
開発：では ^C。そして再び git clone… ゼロからダウンロード始まりましたw

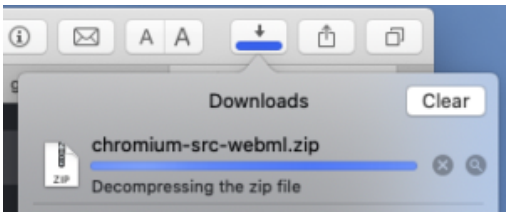
社長：最新版のでなくて構わないんで、tarball になってるやつありますかね。

基盤：ありました。



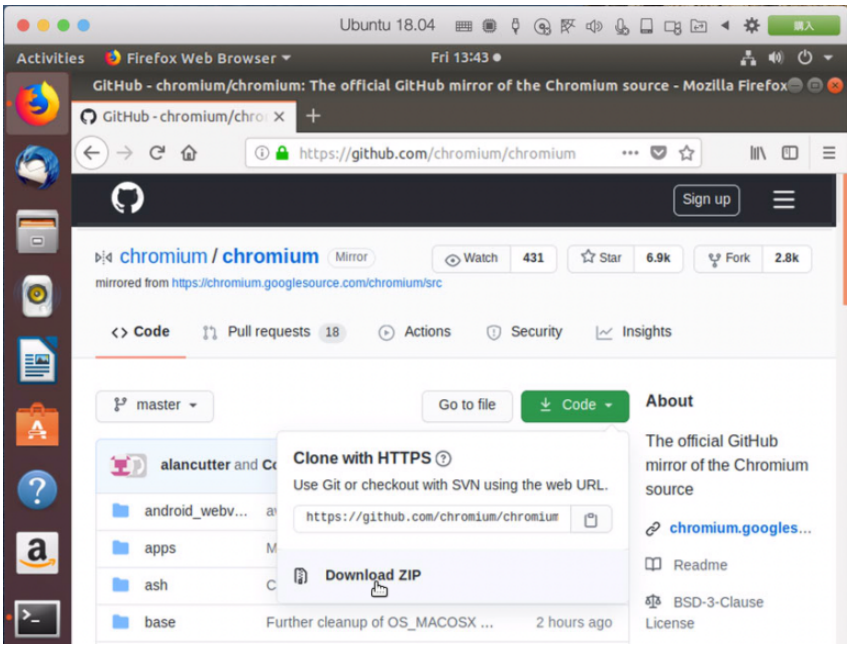
基盤：ダウンロードします。

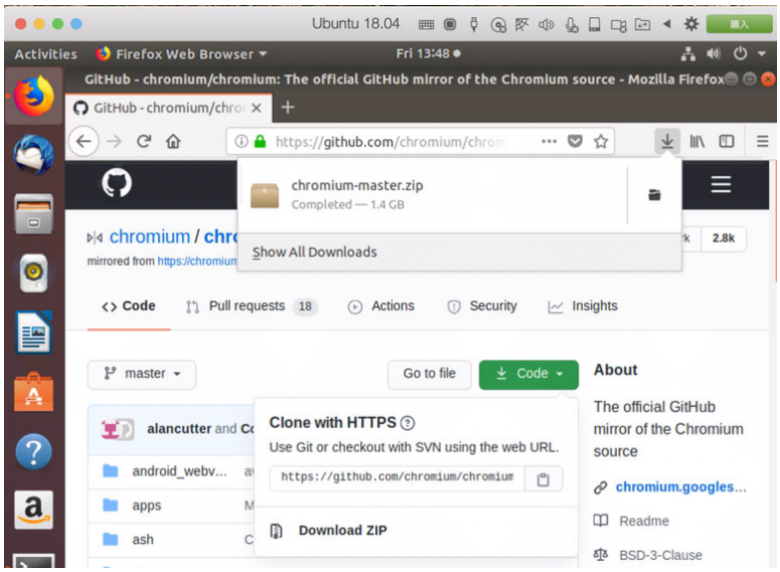




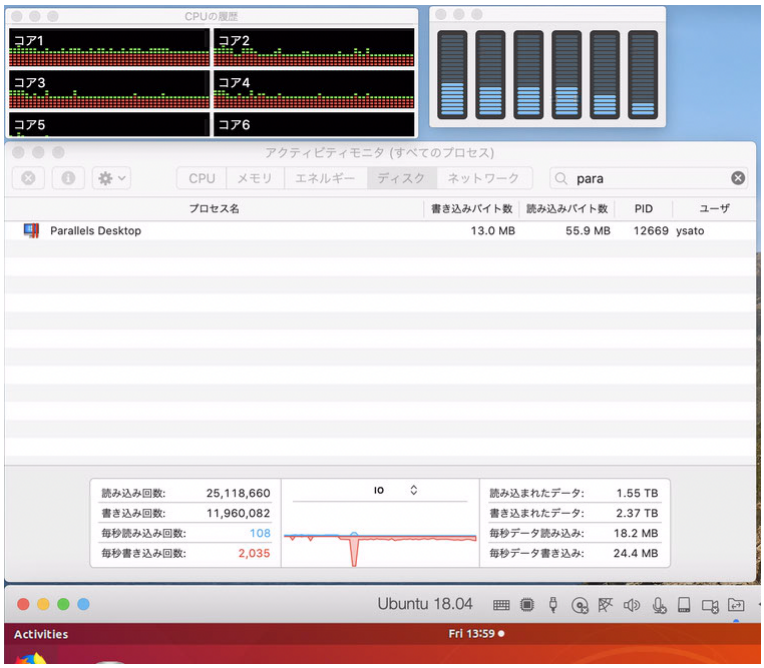
基盤：なんか勝手にdecompressはじめました。それがデフォの設定のようですね。
1.3GB程度の模様。

開発：圧縮の効くソースだと1/10以下になりますからね。


Ubuntu 18.04
Activities
Firefox Web Browser
Fri 13:43
GitHub - chromium/chromium: The official GitHub mirror of the Chromium source - Mozilla Firefox
https://github.com/chromium/chromium
Sign up
chromium / chromium
Mirror
Watch 431
Star 6.9k
Fork 2.8k
mirrored from https://chromium.googlesource.com/chromium/src
Code
Pull requests 18
Actions
Security
Insights
master
Go to file
Code
About
Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/chromium/chromium
Download ZIP
android_webv...
apps
ash
base
Further cleanup of OS_MACOSX ...
2 hours ago
The official GitHub mirror of the Chromium source
chromium.googles...
Readme
BSD-3-Clause License


Ubuntu 18.04
Activities
Firefox Web Browser
Fri 13:48
GitHub - chromium/chromium: The official GitHub mirror of the Chromium source - Mozilla Firefox
chromium-master.zip
Completed — 1.4 GB
Show All Downloads
Code
Pull requests 18
Actions
Security
Insights
master
Go to file
Code
About
Clone with HTTPS
Use Git or checkout with SVN using the web URL.
Download ZIP
The official GitHub mirror of the Chromium source
Readme
BSD-3-Clause

基盤：では、unzip。




```
pu18% ls -l
total 0
-rw-rw-r-- 1 ysato ysato 1471391642 Jul 31 13:46 chromium-master.zip
pu18% time unzip *zip > x

real    5m26.669s
user    0m27.496s
sys     1m55.561s
```

基盤：6分でした。

開発：git でやると3時間。zipでやると10分てことですね。

```
pu18% ls -l
total 0
drwxrwxr-x 1 ysato ysato       2496 Jul 31 12:57 chromium-master
-rw-rw-r-- 1 ysato ysato 1471391642 Jul 31 13:46 chromium-master.zip
pu18% time find | wc
 364007  364137 34090959

real    0m34.383s
user    0m1.782s
sys     0m11.154s
```

社長：巨大なZIPって、一連で圧縮してもそんなに圧縮率も変わらないでしょうし、100MBくらいずつで分割圧縮・並列解凍できると良いのにね。

基盤：このへんのディスクI/Oは、仮想ディスクにしたらもっと速いでしょうね。

```
pu18% time find . | wc
 364009  364139 34090974

real    0m36.841s
user    0m1.562s
sys     0m11.632s
pu18% time du -s .
0       .

real    0m34.948s
user    0m0.864s
sys     0m11.404s
```

基盤：やっぱり、直属の仮想ディスクでやりましょう。最近のLinuxでは超簡単なのはわかっています。parted で拡大して resize2fs で使用。

```
pu18% df /dev/sda1
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       65791532 11302748  51117060  19% /
pu18% sudo resize2fs /dev/sda1
resize2fs 1.44.1 (24-Mar-2018)
Filesystem at /dev/sda1 is mounted on /; on-line resizing required
old_desc_blocks = 8, new_desc_blocks = 16
The filesystem on /dev/sda1 is now 33292032 (4k) blocks long.

pu18% df /dev/sda1
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1      130816068 11310460 113491580  10% /
```

基盤：まず、外部のドライブを使った場合。

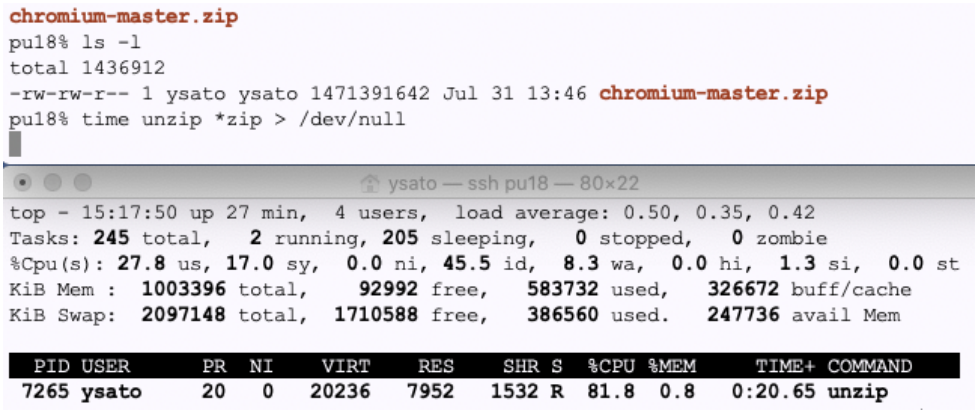
```
pu18% ls -l
total 0
-rw-rw-r-- 1 ysato ysato 1471391642 Jul 31 13:46 chromium-master.zip
pu18% time unzip *zip > /dev/null
```

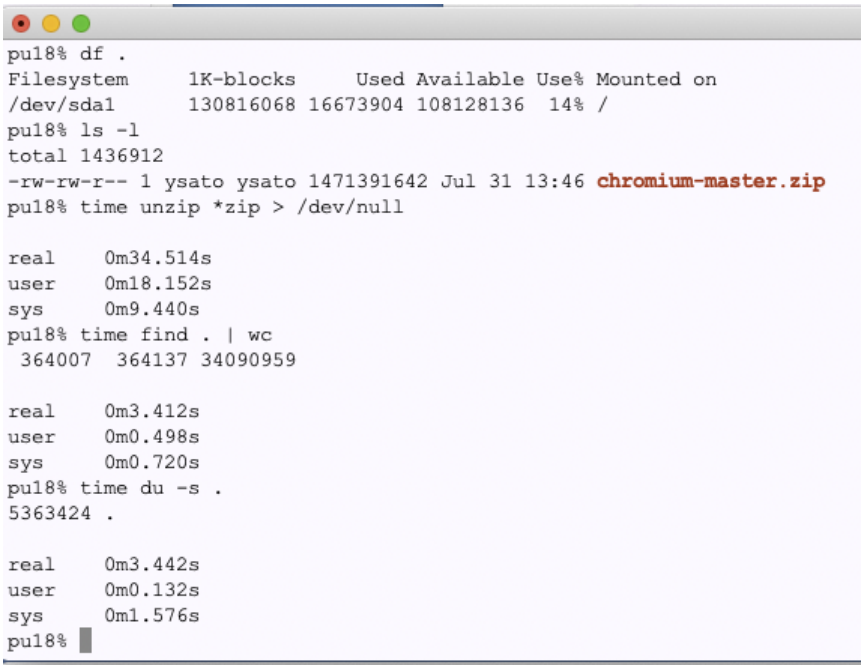
```
ysato — ssh pu18 — 80×22
pu18% pwd
/home/ysato/dev/chromium
top - 15:27:07 up 37 min,  4 users,  load average: 0.28, 0.29, 0.34
Tasks: 245 total,   2 running, 206 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.1 us, 22.4 sy,  0.0 ni, 74.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1003396 total,    81664 free,   528032 used,   393700 buff/cache
KiB Swap:  2097148 total,  1652988 free,   444160 used.   283004 avail Mem
```

```
pu18% df .
Filesystem      1K-blocks      Used  Available Use% Mounted on
Home          2071491588 250931200 1820560388  13% /media/psf/Home
pu18% ls -l
total 0
-rw-rw-r-- 1 ysato ysato 1471391642 Jul 31 13:46 chromium-master.zip
pu18% time unzip *zip > /dev/null

real    5m3.977s
user    0m26.910s
sys     1m43.746s
pu18% time find . | wc
 364008  364138 34090970

real    0m36.467s
user    0m1.517s
sys     0m11.599s
pu18% time du -s .
0       .

real    0m36.697s
user    0m0.825s
sys     0m12.010s
```

開発：CPUは40%程度、ほぼシステム時間。ユーザ時間は3〜5%程度ですね。

基盤：次に、内蔵の仮想ディスクでやった場合。

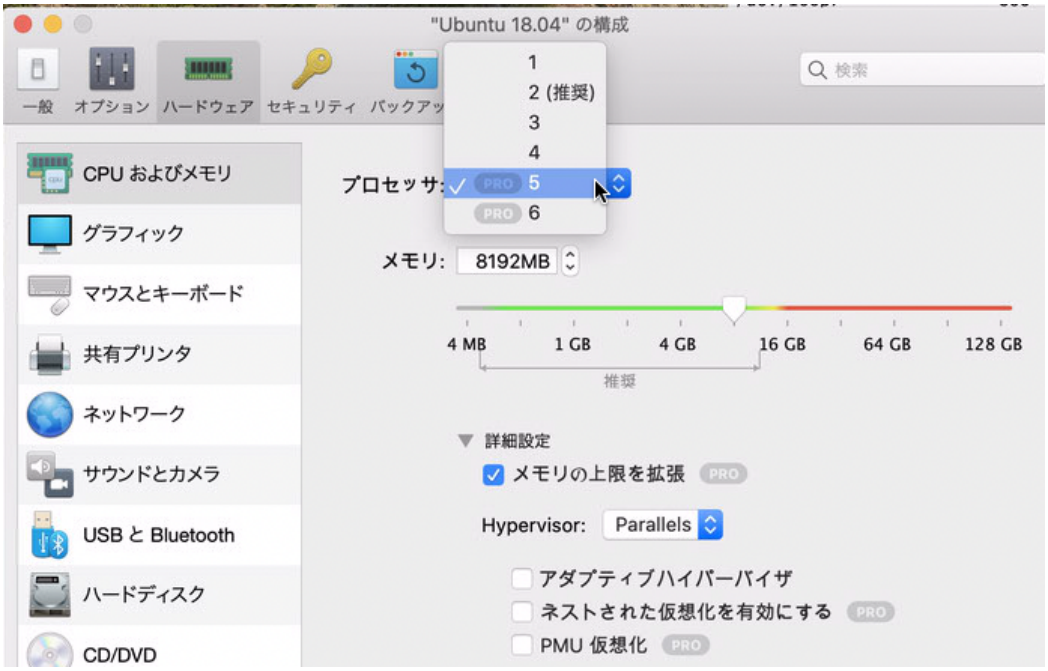




基盤：約**10倍速**と出ました。

開発：うーん、git clone が何時間もかかったのはこのせいですかね…

基盤：あ、あと今気がついたんですが、RAMの割当が1GBなので8GBにします。CPUもデフォが2なのを5に。



開発：これが例の買い切りとサブスクリプションの分水嶺なわけですね。

基盤：ということで、git clone に濡れ衣を着せてしまった感がありますので、正規の手順で再びやります。

社長：しかし、git って、うちみたいにちょっといじってみようかという人の敷居をすごく高くしてますよね。

開発：まああとはただ、たぶん3時間くらい待つだけですか。

経理：アース渦巻第3巻点火しました。

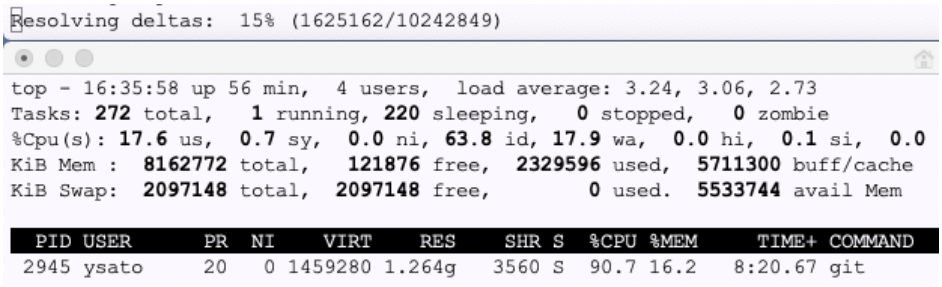
＊　＊　＊

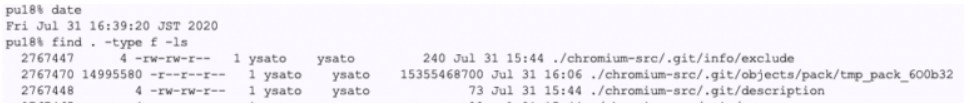
開発：それで shell に戻るんですが、printf があって scanf が無いのは片手落ちだと思うわけです。たとえば ls の結果を分解したい。

```
MacMini% ls -ld Src
drwxr-xr-x@ 211 ysato  staff  6752 Jul 31 15:59 Src
```

開発：それなら scanf “%s %d %s %s %d %t %s” M L U G Z T F とかで分解して、printf “%t %10d %s %s” $T $Z $F `md5 $F` なんてできると便利です。

基盤：ここで残念なお知らせです。作業用のディスクをローカルにうつしても git clone は早くはなりませんでした。ちゃんちゃん。ただ、進捗10%くらいまではCPUは数%で推移していたのですが、それ以降ガツンとCPUを使うことが多くなりました。





開発：何もファイルを作ったり書いたりしている形跡も無いし、ひたすらこの15GBのtmp_pack というのをなめてるんですかね？

基盤：tcpdumpしても、何か通信している気配がないです。

開発：スワップしまくってるようにも見えないですしねえ…

社長：ああそれで、shellのscanfコマンドですが、結果を変数というか、ファイルにも受け取れると良いですね。

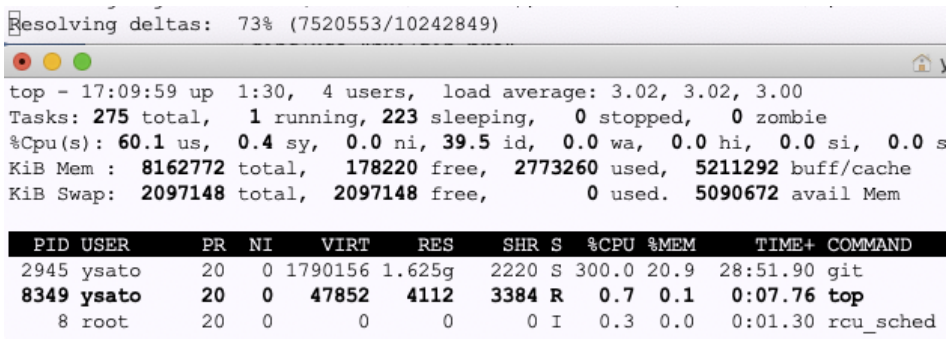
開発：一般化すると、shellの変数は実は全てメモリマップされたファイルである、というのでも良いかなと思います。

社長：ソケットであっても良いかもですね。

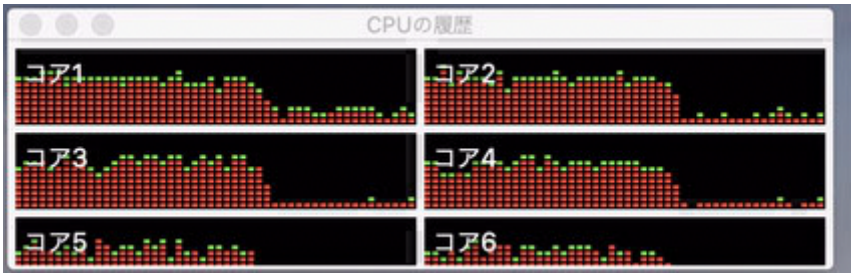
開発：子孫のプロセスから返すデータも、メモリマップされたファイルで受け取れると良いですね。

社長：そのあたり、C言語とかのレベルで、このポインタの先はファイルにあるよというのをサポートしてくれると便利かも知れない。ある種のストレージクラスというか。external で permanent とか sync とか source とか。

基盤：git clone、イキナリ本気モードです。



開発：あたしだってやるときゃやるよみたいな。



基盤：また静かになりました。

開発：しかしこれ、ビルドの様子次第では、Parallels Desktop Pro の線もありですね。少なくとも Acrobat より利用価値が高い。

開発：いやーしかし、この zsh のコーディングスタイルは好きだなー。カッコの前後の空白を別にすれば、ひょっとして自分で書いたのかなーってくらい違和感がありません。

社長：make に awk 使っちゃってたりする所もまた (^-^)

基盤：おっと、git clone 終了しました。所用98分。

```
pu18% time git clone https://github.com/otcshare/chromium-src.git
Cloning into 'chromium-src'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 12999042 (delta 2), reused 2 (delta 2), pack-reused 12999039
Receiving objects: 100% (12999042/12999042), 14.30 GiB | 11.21 MiB/s, done.
Resolving deltas: 100% (10242849/10242849), done.
Checking out files: 100% (315103/315103), done.

real    98m24.808s
user    34m43.075s
sys     2m54.226s
```

社長：それはそうと、ls こそ shell の組み込みコマンドであるべきかなと思うんですが。というか、ファイルのメタ情報をフォーマットして出力する機能が。基盤：それは ls というより stat コマンドみたいですね。

開発：え、このコマンドいつからあったんですかね。man stat、HISTORY … The **stat** utility appeared in NetBSD 1.6 and FreeBSD 4.10. … BSD May 8, 2003 BSD。へー。

開発：ls の出力を分解して使用するという利用例は、意味がなくなりましたｗ。こういう感じですかね…

```
MacMini% find . -exec stat -t '%Y-%m-%d:%H:%M:%S' -f '%Sa %8z %N' '{}' ';'
2020-07-14:12:09:53     1408 .
2020-07-13:05:08:13      384 ./Misc
2020-07-13:05:08:13     6499 ./Misc/globtests
```

開発：便利ですね。

＊　＊　＊

開発：ディレクトリのインデックスHTML生成ツールの原型はできたのですが、Macの上で出来たんで Ubuntu に持ってったら動かないんです。stat というコマンド名なんですが、全然仕様も由来も違うんですね。Macのほうが正統なBSD流。Ubuntuのはおもちゃみたいの。

社長：やはり shell の組み込みにしちゃって、その shell だけ持ち歩けばどこでもOKっていうのが良いかもですね。

開発：出力の整数を引き算する必要があったのですが、これも　shell でできないから、久しぶりに dc なんて使っちゃいました。

社長：ポーランド記法！

開発：これもまた、shell に組み込みだったら良いのにと思いましたね。

社長：ツールを作る人は、それが最終の製品だと思って作ることが多いからUIとかファイルI/Oとかも含めて内部依存的にできちゃってて、さらにそれをライブラリとして使う時に難儀しますね。

開発：なので、そのあたりを切り出して、動的ライブラリ化するというのも面白いのでは無いかと思うんです。おそらくライセンス上も問題無い。MozillaやChromeだって、ライブラリにして簡単に使えたら、面白い応用があると思うんです。

社長：ツール・アプリとライブラリの境目は面白いですね。密接に結合するより Unix のパイプでつなぐ式から始まったデータだけで結合したほうが組み合わせてきには面白い。けれど性能的にはデメリットはある。標準化もされにくい。

開発：ライブラリのAPIとか通信プロトコルだと仕様がかっちり決められて互換性が維持されますけど、コマンドの出力データって人間が読めればいいやというか人間の読みやすさに合わせて設計されているから、標準化されにくいですね。

社長：そのあたりもひっくるめて、何をどうすると幸せかを考えていきましょう。

開発：書きなぐりですが、今日書いた部分をアーカイブして置きます。

▶ dir2html.sh

＊　＊　＊

基盤：結局ParallelsでのChromeのビルドは所要2時間56分（176分）でした。

```
+ time autoninja -C out/Default chrome unstable_deb
ninja: Entering directory `out/Default'
[42457/42457] STAMP obj/chrome/installer/linux/unstable_deb.stamp
49617.42user 2741.93system 2:56:02elapsed 495%CPU (0avgtext+0avgdata 1830780maxresident)k
2789560inputs+8444008outputs (13200major+1077143894minor)pagefaults 0swaps
```

基盤：同じ版のautoninja がHyper/VのLinuxで3時間40分（220分）でしたが、あれはディスクをQNAPにNFSマウントしていたことによる影響が大きいと思われます。

開発：コンパラブルな性能ということですかね。使い勝手的には、Windows 10 の上のHyper/V のLinux VMをMacからリモートデスクトップで覗くより、Mac自身の上のParallelsでLinux VMで直接いじれたほうが便利ですが。

基盤：実用という意味では、Linux VMの仮想デスクトップをMacの上で直接表示すれば、あまり使い勝手に違いが無いとは思いますけどね。ただ、LinuxのVNCサーバがちゃんと使えるのかというところに課題が。

開発：まああれも、ちゃんとRealVNCにお金を払えばよいのでしょうけどね。

経理：RealVNCは4月に検討しましたね。$4.59/月@1サーバ。1LSUです。

基盤：リーズナブルっちゃリーズナブルですかね。

社長：経費的には問題ないと思いますが、そもそも X Window のほうが良くね？と思うのですが、どうなんでしょう？

開発：本当に更新が必要な情報だけを転送しているわけですしね。情報の作り手が使い手にダイレクトに情報を送る。中間で推測し直したりしない。

基盤：Chromiumって X Window 対応してるんですかね？

— 2020-0731 SatoxITS