

rsync によるバックアップ

<http://x68000.q-e-d.net/~68user/unix/pickup?rsync>

<http://www.itmedia.co.jp/enterprise/articles/0804/25/news034.html>

tar を使う方法とかあるけど、rsync もなかなか良い。

オプションの -a はアーカイブモードで -rlptgoD と同じである。各オプションの意味は以下のとおり。

ディレクトリを再帰的にコピーする (-r)
シンボリックリンクを、そのままシンボリックリンクとしてコピーする (-l)
パーミッションをそのままコピーする (-p)
タイムスタンプをそのままコピーする (-t)
グループをそのままコピーする (-g)
ファイルオーナー (所有者) をそのままコピーする (-o)
デバイスファイルや特殊ファイルを、そのままコピーする (-D)

```
rsync -av /foo/from_dir /bar/to_dir
```

ssh サーバに転送するには

```
rsync -av -e ssh user@example.com:from_dir/ /foo/to_dir/
```

ファイル名の文字コードを変換しながら転送

<http://rottarte.net/blog/archives/2847>

```
--iconv=SRC-CHARSET,DEST-CHARSET
```

というオプションを使うとファイル名の文字コードを変換できる。

```
rsync -auv --iconv=EUC-JP,UTF8 ./ 123.123.123.123 :/dest/dir/
```

差分バックアップ

差分バックアップするには --link-dest オプションを使う

```
rsync -av --delete --link-dest=../dir1_bak0 ^/dir1/ ^/dir1_bak1
```

相対パスを利用する場合は、--link-dest で指定するディレクトリは、コピー先ディレクトリからの相対パスになるの注意。

絶対パスを使うと安心。

--link-dest は、このオプションで指定したディレクトリ以下とコピー元を比較し、ファイルの所有者、タイムスタンプ、パーミッションなどすべてが一致するファイルであれば、指定されたディレクトリ内のファイルのハードリンクを作成します (ファイルが一致しない場合は、ハードリンクではなくコピーが行われます)

```
ls -li
```

でハードリンクになっていることが確認できる。

■ サンプルシェル

```
#!/bin/sh
export LANG=ja_JP.UTF-8

#####
# 環境変数
#####
nowTime=`date +%Y%m%d_%H%M%S`
limit=5
sourcePath="/home/centos/temp/backup/src"
backupBasePath="/home/centos/temp/backup/dest"
destPath=${backupBasePath}/${nowTime}
workPath=${backupBasePath}/work

execListBackupBasePath="ls ¥¥${backupBasePath}¥¥ | grep ¥"[0-9]¥{8¥}_[0-9]¥{6¥}¥""
pidFile=/var/run/backup.pid
mail="hoge@hoge.com"
subject="[Backup Log] ${nowTime}"

# LINE_SEP='
# '
LINE_SEP=$'¥n'
IFS_DEFAULT=$IFS

#####
# 関数群
#####
# sendMail(){
#   subjectBase64=""
#   lines=`echo -e ${subject} | fold -c10`
#   sep=""
#   IFS=${LINE_SEP}
#   for line in ${lines}; do
#     subjectBase64=${subjectBase64}${sep}"=?UTF-8?B?`echo -e ${line} | base64`?="
#     sep="¥n¥t"
#   done
#   IFS=${IFS_DEFAULT}
#   #subject="${subjectBase64%¥n¥t}"
#   # echo -e ${subjectBase64}
#   body=`echo -e $1 | base64`
#   echo -e "To:${mail}¥nSubject:${subjectBase64}¥nContent-Type: text/plain; charset=UTF-8
¥nContent-Transfer-Encoding: base64¥n¥n$body" | /usr/sbin/sendmail -t
# }

checkError(){
  if [ $1 -ne 0 ]; then
    echo $2
    # sendMail " バックアップ異常終了 ¥n$2"
    exit $1
  fi
}

checkProcess(){
  if [ -f "${pidFile}" ]; then
    PID=`cat "${pidFile}"`
    if (ps -e | awk '{print $1}' | grep ${PID} > /dev/null); then
      return 1
    fi
  fi
  echo $$ > "${pidFile}"
  return 0
}

judgeFullBackup() {
  local result=0
  fullBackup=1
  local countBackup=`eval ${execListBackupBasePath} | wc -l`
  if [ ${countBackup} -ne 0 ]; then
    fullBackup=0
    beforeBackup=`eval ${execListBackupBasePath} | sort -r | head -n 1`
    result=$?
    beforeBackup="${backupBasePath}/${beforeBackup}"
  fi
  return ${result}
}
```

```

}

runBackup() {
    if [ -d "${workPath}" ]; then
        echo 作業用ディレクトリを削除します。
        echo rm "${workPath}"
        rm -rf "${workPath}"
        result=$?
        checkError ${result} " 作業用ディレクトリ削除に失敗しました。 "
    fi

    if [ $fullBackup -eq 1 ]; then
        echo バックアップ種類 ベースバックアップ
        echo バックアップソース "${sourcePath}"
        echo バックアップ先 "${destPath}"
    else
        echo バックアップ種類 差分バックアップ
        echo バックアップソース "${sourcePath}"
        echo バックアップ比較元 "${beforeBackup}"
        echo バックアップ先 "${destPath}"
    fi
    echo ${LINE_SEP}

    if [ $fullBackup -eq 1 ]; then
        #(time rsync -av --chmod=ugo-w --delete "${sourcePath}/" "${workPath}/")
        (time rsync -av --delete "${sourcePath}/" "${workPath}/")
    else
        #(time rsync -av --chmod=ugo-w --delete --link-dest="${beforeBackup}" "${sourcePath}/"
"${workPath}/")
        (time rsync -av --delete --link-dest="${beforeBackup}" "${sourcePath}/" "${workPath}/")
    fi
    result=$?
    checkError ${result} " バックアップに失敗しました。 "

    mv "${workPath}" "${destPath}"
    result=$?
    checkError ${result} " 作業ディレクトリの名前変更に失敗しました。 "

    return 0
}

deleteBackup() {
    IFS=$LINE_SEP
    local countBackup=`eval ${execListBackupBasePath} | wc -l`
    local countDelete=`expr ${countBackup} - ${limit}`
    if [ $countDelete -gt 0 ]; then
        local deleteDirs= `eval ${execListBackupBasePath} | sort | head -n ${countDelete}`
        for deleteDir in ${deleteDirs}; do
            deleteDir="${backupBasePath}/${deleteDir}"
            echo delete "${deleteDir}"
            (time rm -rf "${deleteDir}")
        done
    fi
    IFS=${IFS_DEFAULT}
}

#####
#   メイン処理
#####
echo "#####"
echo backup start
date "+%Y-%m-%d %H:%M:%S"
echo "#####"

checkProcess
result=$?
checkError ${result} " 多重起動エラー "

# パスの存在チェック
if [ ! -d "${sourcePath}" ]; then
    checkError 1 "${sourcePath} がありません。 "
fi

if [ ! -d "${backupBasePath}" ]; then
    checkError 1 "${backupBasePath} がありません。 "
fi

if [ -d "${destPath}" ]; then
    checkError 1 "${destPath} が既に存在しています。 "
fi

# バックアップモード決定

```

```

judgeFullBackup
result=$?
checkError ${result} " バックアップモード決定に失敗しました。 "

# バックアップ実行
runBackup
result=$?
checkError ${result} " バックアップに失敗しました。 "

# 古いバックアップファイルを削除
deleteBackup

echo "#####"
echo backup end
date "+%Y-%m-%d %H:%M:%S"
echo "#####"
rm "${pidFile}"

# sendMail " バックアップ正常終了 "
echo " バックアップ正常終了 "

```

cron を使う場合の注意

cron で実行すると LANG 環境変数が設定されずにログファイル等が化けることがある。
cron から実行されるシェル等の冒頭で

```
export LANG=ja_JP.UTF-8
```

で文字コードを設定しておくが良い。

ファイルの変更検出について

rsync は通常のオプションでは、日付とサイズによってファイルの変更検出を行う。
ファイルの中身で変更検出を行う場合は
-c または --checksum

転送要否を決定する際、タイムスタンプとファイルサイズではなく、128bit の MD4 チェックサムを用いて同一ファイルか否かをチェックする。

を使う。

バックアップ領域の書込権限

可能であれば、バックアップ領域は読取専用にする。
通常は root 権限でも書込出来ないように読取専用でマウントしておく。
バックアップ時に書込み可能にする。

■ 方法 1

通常は

```
mount -o ro /dev/sdb1 /mnt/backup
```

にしておき、バックアップ時に

```
mount -o remount,rw /mnt/backup
```

にする。ただし、バックアップ中は書込み可能になる

■ 方法 2

名前空間を使ってバックアッププロセス以外は書込不可にする。
通常は

```
mount -o ro /dev/sdb1 /mnt/backup
```

にしておく。バックアッププロセスを

```
unshare -m
```

で実行して分離している状態で

```
mount -o remount,rw /mnt/backup
```

にする。