目的

選択したフォルダの配下の、各フォルダのディスク使用量を調べて、結果をCSVファイルに保存します

機能

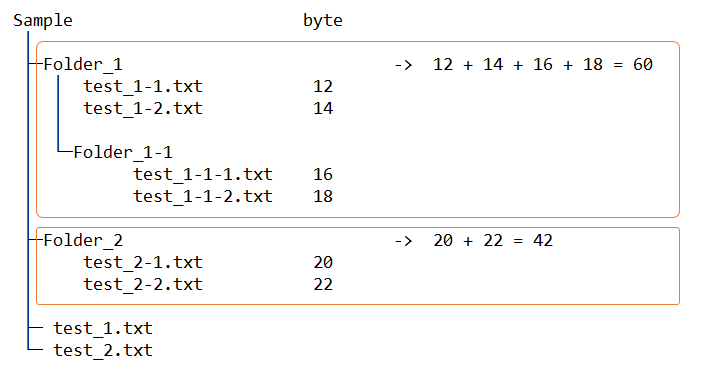
下図の枠内のファイル容量を集計し、CSVに書き出します

- 選択したフォルダの配下の、各フォルダ内のファイル容量を、ツリーに沿って合計していきます

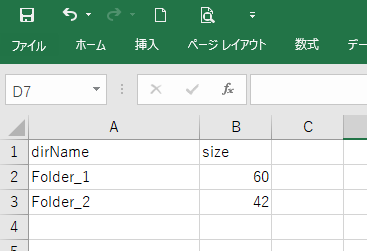
- 選択したフォルダの直下のフォルダごとに集計します　（Folder\_1、 Folder\_2　ごとに集計）

- 選択したフォルダの直下のファイル容量は無視します　（test\_1.txt、 test \_2.txt　は集計しない）

Sampleフォルダを選択した場合

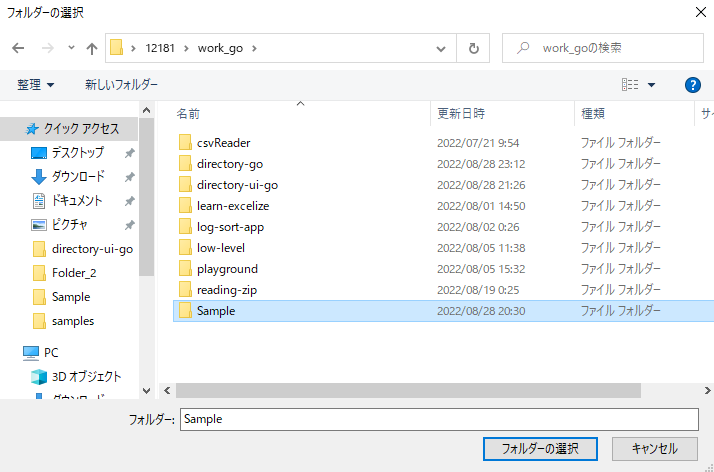


出力されるCSV

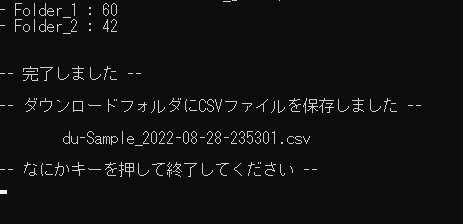


使い方

1. directory-go.exeを、デスクトップなどの適宜な場所にコピペする　（ショートカットではなく）
2. directory-go.exeを、ダブルクリックする
3. ダイアログが表示されるので、フォルダを選択する



1. 処理が始まるのでしばらく待つ　（フォルダの階層が深い場合は時間がかかる）
2. 「完了しました」の表示が出たら、なにかキーを押して終了する



1. 自分のダウンロードフォルダに、　du-<選択したフォルダ名>\_<日付時刻>.csv　が保存される

注意

実行ユーザーがアクセス権を持たないフォルダは、アクセスできないので計算ができません

コンソールにエラーが表示されます　（フォルダの数が多い場合は一瞬で上のほうにスクロールされる）

参考

環境



サードパーティーのパッケージ

- ncruces/zenity　　ダイアログを表示させるために使用

ソースコード

package main

import (

    "bufio"

    "encoding/csv"

    "fmt"

    "io/ioutil"

    "log"

    "os"

    "path/filepath"

    "strconv"

    "time"

    "github.com/ncruces/zenity"

    "golang.org/x/text/encoding/japanese"

    "golang.org/x/text/transform"

)

func main() {

    const defaultPath = ``

    path, err := zenity.SelectFile(zenity.Filename(defaultPath), zenity.Directory())

    fmt.Println(path)

    data := [][]string{{"dirName", "size"}}

    dirs, err := ioutil.ReadDir(path)

    if err != nil {

        log.Fatal(err.Error())

    }

    for \_, f := range dirs {

        if f.IsDir() {

            var p = filepath.Join(path, f.Name())

            size, err := DirSize(p)

            if err != nil {

                fmt.Println(err.Error())

            }

            fmt.Printf("- %s : %d\n", f.Name(), size)

            sl := []string{f.Name(), strconv.FormatInt(size, 10)}

            data = append(data, sl)

        }

    }

    hd, err := os.UserHomeDir()

    if err != nil {

        log.Fatal(err.Error())

    }

    fn := "du-" + filepath.Base(path) + "\_" + time.Now().Format("2006-01-02-150405") + ".csv"

    fp := filepath.Join(hd, "Downloads", fn)

    f, err := os.Create(fp)

    if err != nil {

        log.Fatal(err.Error())

    }

    defer f.Close()

    w := csv.NewWriter(transform.NewWriter(f, japanese.ShiftJIS.NewEncoder()))

    w.WriteAll(data)

    w.Flush()

    if err := w.Error(); err != nil {

        log.Fatal(err)

    }

    fmt.Print("\n\n")

    fmt.Println("-- 完了しました --")

    fmt.Print("\n")

    fmt.Println("-- ダウンロードフォルダにCSVファイルを保存しました --")

    fmt.Print("\n")

    fmt.Printf("        %s\n\n", fn)

    fmt.Println("-- なにかキーを押して終了してください --")

    scanner2 := bufio.NewScanner(os.Stdin)

    scanner2.Scan()

}

func DirSize(path string) (int64, error) {

    var size int64

    err := filepath.Walk(path, func(\_ string, info os.FileInfo, err error) error {

        if err != nil {

            return err

        }

        if !info.IsDir() {

            size += info.Size()

        }

        return err

    })

    return size, err

}