PS2 – PROGRAMÁTORSKÁ DOKUMENTACE

PS2plotter

Autor: Tomáš Jelínek

**Popis skriptu**

Skript PS2plotter, který slouží ke spouštění celého programu se skládá z hlavního skriptu PS2plotter a 4 knihoven lib.sh, parseparameters.sh, processdata.sh a generateframes.sh. Knihovnami zde nazývám samostatně nespustitelné soubory, obsahující funkce volané z hlavního skriptu.

Využití funkcí je zde primárně navrženo tak, aby zpřehledňovalo zdrojový kód skriptu a rozdělovalo jej do logických částí řešící danou problematiku. Funkce proto nejsou většinou navrženy tak, aby byly univerzální, ale využívají globálně nastavené proměnné, které dále pozměňují. Proměnné, které by bylo možno označit jako klíčové a které se používají napříč funkcemi zapisuji velkými tiskacími písmeny.

**Běh skriptu**

1. Po startu skriptu načtu pomocí příkazu source knihovny do hlavního skriptu.
2. Otestuji, zda jsou nainstalované externí programy jako např: gnuplot, ffmpeg …
3. Dále pomocí funkce setVariables nastavím klíčové proměnné na defaultní hodnoty, tak aby byly připraveny na naplnění.
4. Funkcí parseParameters zpracuji přepínače z příkazové řádky do v předchozím kroku připravených proměnných. Pro parsování řádky využívám nástroj getopt a tudíž jsem schopen zpracovat i dlouhé přepínače.
5. Pokud byl zadán platný konfigurační soubor, tak zavolám funkci parseconfig, která zpracuje jednotlivé direktivy do proměnných z kroku 3.
6. Funkce examineData pak testuje, zda zadané soubory s daty nejsou prázdné a dají se číst. Pokud ne vyzkouší, zda se nejedná o URL. Stáhne obsah platných internetových zdrojů do dočasného souboru vytvořeného pomocí mktemp a uloží si jméno tohoto souboru. Pokud se zjistí že se nejedná o soubor ani o platnou URL pak je zavolána funkce err a skript ukončen. Výsledkem je v proměnné uložený seznam platných souborů s nějakými daty.
7. Pomocí funkce mergeVariables určím výsledné hodnoty vstupních proměnných zadaných uživatelem a určím zda jsou platné. Přednost mají vždy hodnoty zadané přepínači, dále konfiguračním souborem a nakonec defaultní hodnoty. Pokud není nastavena možnost Ignore Errors pak, při nalezení chybné hodnoty ukončím skript chybou. V opačném případě vypíši jen varování a pokusím se nahradit chybnou hodnotu hodnotou nižší priority. Až pokud ani tak neuspěji tak skončím chybou.
8. Dále otestuji soubory s daty, jestli obsahují pouze validní hodnoty času a hodnoty typu int nebo float. Pokud ne tak na chybový výstup vypíši jméno souboru a číslo nevalidního řádku a skončím.
9. Pokud je zadáno více souborů setřídím, soubory podle v nich uložených časů, tak že soubor s nejnižším časem na prvním řádku je první a soubor s nejvyšším na posledním. Pro řazení využívám algoritmus Bubblesort, neboť předpokládám počty souborů v řádech jednotek, maximálně desítek. Jestliže se po seřazení soubory překrývají vypíši chybu a skončím.
10. Následně ve funkci prepareData vytvořím dočasný soubor pro plot v gnuplotu, který obsahuje navíc sloupec s vypočtenou hodnotou barvy efektu. Pokud je nastaveno xmin nebo xmax na hodnotu jinou než auto, použiji pouze řádky s časy ležícími mezi těmito hodnotami.
11. Když mám data zpracovány můžu dopočítat potřebné Time,Speed,FPS, protože už znám počet řádků s daty. Nastavím si také hodnoty rozsahů plotu xmin, xmax, ymin, ymax. Připravím si táké řetězec se set příkazy pro gnuplot, které zadal uživatel jako modifikaci.
12. Ve funkci generateFrames potom pomocí gnuplotu vytvořím jednotlivé snímky, ze kterých pomocí programu ffmpeg sestavím výsledné video ve formátu mp4.
13. Na závěr odstraním všechny dočasné soubory a skončím

**Knihovna: lib.sh**

**max\_dir()**

Vrací číslo+1 posledního adresáře v danem adresáři, oddělovačem je \_

PARAMETRY

$1 - prohledávaný adresář, $2 - jméno adresáře

**control\_c()**

Zavolána, pokud nastane přerušení ctrl+c. Zajistí úklid a ukončení skriptu.

**cleanAll()**

Smaže dočasný adresář a soubory.

**err()**

Vypíše chybovou hlášku a ukončí skript s návratovým kódem 1.

**warn()**

Pokud je program v režimu verbose, tak vypíše na chybový výstup varování.

**isIntFloat()**

Testuje zda je parametr $1 hodnota typu int nebo float. Pokud ano vraci 1 jinak 0

PARAMETRY

$1 – testovana hodnota

**testFile()**

Otestuje zda soubor existuje, lze jej číst a není prázdný. Pokud to splňuje vrací 1 jinak 0.

PARAMETRY

$1 – cesta k souboru

**TStampMatch()**

Otestuje, zda časový řetězec odpovídá TimeStamp. Pokud to splňuje vrací 0 jinak 1.

PARAMETRY

$1 - časový řetězec, $2 - Timestamp

**Strptime()**

Otestuje, zda časový řetězec odpovídá TimeStamp. Vrací časový řetězec, jestliže odpovídá, jinak prázdný řetězec

PARAMETRY

$1 - časový řetězec, $2 – Timestamp

**isbiggerDate()**

Otestuje, zda časový řetězec2 je větší než časový řetězec1. Vrací True jestliže řetězec2 > řetězec1 jinak False.

PARAMETRY

$1 - časový řetězec1, $2 - časový řetězec2, $3 – Timestamp

**inSeconds()**

Převede časový řetězec na sekundy od počátku strojového času

PARAMETRY

$1 - časový řetězec, $2 - Timestamp

**Knihovna: parseparameters.sh**

**parseconfig()**

Slouží ke zpracování konfiguračního souboru. Do proměnných definovaných ve funkci setVariables uloží obsah direktiv z konfiguračního souboru

PARAMETRY

$1 - jméno konfiguračního souboru

**parseParameters()**

Zpracovává přepínače z příkazové řádky do proměnných definovaných ve funkci setVariables. Pro zpracovaní využívá getopt, tudíž podporuje rozpoznávání krátkých i dlouhých přepínačů.

PARAMETRY

$@ - obsah příkazové řádky

**usage()**

Vypisuje návod na používání skriptu.

**printVariables()**

Vypíše hodnoty proměnných, které zadal uživatel. Slouží k ověření konfigurace skriptu.

**setVariables ()**

Nastaví proměnné na defaultní hodnoty na začátku běhu skriptu.

**mergeVariables()**

Sloučí hodnoty proměnných z přepínačů a konfiguračního souboru do výsledné hodnoty tak, že přednost mají přepínače. Pokud není nastavena proměnná IE, která říká, že se mají ignorovat nekritické chyby, tak je při zjištění nevalidní hodnoty zavolána funkce err s příslušnou hláškou. V opačném případě se ještě zkouší, jestli chybná hodnota z parametru nemůže být opravena hodnotou z konfiguračního souboru nebo defaultní hodnotou a je zavolána funkce warn s příslušnou hláškou.

**Knihovna: processdata.sh**

**examineData ()**

Funkce testuje, zda zadané soubory jsou neprázdné soubory s právy pro čtení. Pokud ne vyzkouší, zda se nejedna o URL. Stáhne obsah platných internetových zdrojů do souboru a uloží si jméno tohoto souboru. Pokud se zjistí že se nejedná o soubor ani o platnou URL pak je zavolána funkce err.

***testDataValidity ()***

Otestuje, zda všechny datové soubory obsahují pouze validní data.

**sortFiles ()**

Podle časových údajů setřídí pořadí od souborů s nejmenšími časy až ty po největší. Pro řazení je použit algoritmus Bubblesort, neboť se zde předpokládají počty řazených souborů v řádech jednotek, maximálně desítek. Pokud se data v souborech překrývají zavolá funkci err.

**prepareData()**

Vytvoří soubor s daty pro plot, tento soubor obsahuje další sloupec určující barvu efektu.

**computeTimeSpeedFps()**

Funkce dopočítá potřebné hodnoty time,speed a fps.

**computeMINMAX()**

Dopočte hodnoty xmin,xmax,ymin,ymax pokud jsou zadány jako min nebo max

**Knihovna: generateframes.sh**

**computePlot()**

Nastavení hodnot proměnných potřebných pro běh gnuplotu.

**makeSets()**

Uloží řetězec s nastavením rozsahu plotu, kritických hodnot a parametry gnuplotu do proměnné tak, aby je bylo později možno gnuplotu snadno předat.

**generateFrames()**

Pomocí gnuplotu vytvoří jednotlivé snímky animace. Z těchto snímků je následně vytvořeno video ve formátu mp4.