IB016 – Druhý domácí úkol – práce s IO

Termíny

- **10. května 2020** (vč.) pro implementační část.
- **17. května 2020** (vč.) pro peer review.

Druhá úloha bude zaměřena na především na práci se souborovým systémem. Vaším úkolem bude naprogramovat program, který dokáže uspořádat fotografie z různých adresářů do adresářů podle událostí.

Úloha je trochu volnější. Kromě základní funkcionality budete mít za úkol implementovat dvě z několika nabízených drobných rozšíření. Tentokrát neexistuje předepsaná kostra.

Motivace

Na různých událostech se obvykle fotí víc než jedním fotoaparátem/mobilem. Jedním zařízením se naopak fotí více než jedna událost. Při organisaci dostanete fotky rozdělené podle zařízení a jako výsledek chcete fotky roztříděné podle událostí, a to navíc v rámci každé události očíslované podle času vzniku.

Základní část

Poznámka: v zadání budeme používat podobné značení, s jakým se můžete setkat např. v manuálových stránkách. Výsledný nástroj zde bude pojmenován `./tidy`.

`./tidy [-h | --help]`

Bez parametrů či s přepínačem `-h` nebo `--help` nástroj vypíše nápovědu, v níž bude shrnuto, jak se dá volat a jaké volby rozeznává. Pro inspiraci zkuste spustit s `--help` nějakou základní řádkovou utilitu (třeba `cat`).

`./tidy [VOLBY] SRC_DIR [DEST_DIR]`

Roztřídí fotografie ze `SRC_DIR` (zanořené libovolně hluboko, s výjimkou `.nomedia` níže) do `DEST_DIR` (není-li zadáno, použije se současný adresář). Za fotografie pro jednoduchost považujeme všechny soubory s koncovkou `.jpg` a `.jpeg` bez ohledu na velikost písmen. V adresářích se můžou nacházet i jiné soubory, které nesmí být změněny či přesunuty.

Narazí-li program při prohledávání na adresář, v němž existuje soubor `.nomedia`, nepřesouvá z tohoto adresáře ani jeho podadresářů žádné fotografie.

V `SRC_DIR` se musí nacházet soubor s názvem `tidy.conf`, jehož formát je popsaný v sekci níže. Ten předepisuje, jak budou fotografie na základě času změny souboru přesunuty do podadresářů v `DEST_DIR`. Nepřísluší-li čas fotografie žádnému záznamu v `tidy.conf`, použije se podadresář `other`.

Přesunuté fotografie nástroj navíc očísluje podle času, tj. na začátek jména přidá pořadové číslo. Pořadová čísla tvoří v rámci jednoho adresáře postupku, a aby se podle nich dalo řadit, nízká čísla musí být zleva zarovnaná nulami. Seřazení podle názvu pak musí odpovídat seřazení podle času fotografie.

Cílové adresáře událostí, do nichž nepatří žádné fotografie, se nevytvoří.

Formát konfiguračního souboru

Každý neprázdný řádek, který není komentářem, předepisuje interval od (včetně) a do (nevčetně) času změny souboru a (pod)adresář (zbytek řádku), do něhož se mají fotografie spadající do intervalu přesunout. Například:

```
# Řádky začínající mřížkou a prázdné řádky se ignorují

2020-01-03 08:24 2020-04-03 19:35 Norsko
2020-04-03 19:35 2020-04-04 02:00 Norsko/zpatecni_cesta
2020-04-04 03:20 2020-04-11 18:15 karantena
```

Tento formát je minimum, které program musí podporovat; při parsování data a času však můžete být libovolně benevolentní. Můžete předpokládat, že čas začátku je ostře menší než konce a intervaly se nepřekrývají.

Volby

V základu musí program kromě voleb pro výpis nápovědy podporovat pouze jednu dvojici přepínačů:

- `-n`, `--dry-run`: neprovádí žádné změny; pouze vypíše, které adresáře by se vytvořily a které soubory by se kam přesunuly (vč. změny názvu). Tato volba se hodí pro testování a pro kontrolu před provedením potenciálně destruktivní operace.

Některá rozšíření spočívají v implementaci dalších voleb. Můžete počítat s tím, že všechny volby budou uvedeny na začátku, ale váš program může být při čtení argumentů libovolně benevolentní (např. rozpoznávat volby i na konci či podporovat klasické shlukování krátkých přepínačů).

Základní parsování argumentů nemusí být nikterak sofistikované, ale pokud budete chtít implementovat některé z „přepínačových“ rozšíření, může se hodit nějaký modul specialisovaný na práci s argumenty příkazové řádky. Jeden takový je přímo v balíku `base`.

Rozšíření

Kromě základu výše musí váš program obsahovat alespoň dvě z následujících rozšíření. Zvolená rozšíření prosím napište do komentáře na začátku souboru.

Mělo by být pohodlně proveditelné nejdříve naimplementovat základ a až poté rozšíření – tj. rozšiřující funkcionalita by neměla vyžadovat zásadní překopání celého základu. I tak je ale vhodné při psaní základu uvažovat nad dalším postupem – např. do kterých pomocných funkcí se hodí předávat konfiguraci.

1. Více zdrojů

Podobně jako vícezdrojové `cp` a `mv`:

```
./tidy [VOLBY] SRC_DIR
./tidy [VOLBY] SRC_DIR... DEST_DIR
./tidy [VOLBY] -t DEST_DIR SRC_DIR...
```

Konfigurační soubor v každém zdrojovém adresáři ovlivňuje pouze daný adresář.

2. Konfigurovatelnost

Podpora pro další volby:

- `-f SOUBOR` – použije `SOUBOR` místo `SRC_DIR/tidy.conf`,
- `-v [N]`, `--verbose [N]` – úroveň upovídanosti (např. 1 = vypisuje, co dělá, 2 = vč. ladicích informací, např. načtené intervaly, výsledná konfigurace apod.), není-li zadána, předpokládá se 1,
- `-q`, `--quiet` – jako `-v 0`, krom chyb nehlásí nic (výchozí).

Platí poslední z `-v` a `-q`; více konfiguračních souborů zadaných přes `-f` se kombinuje. Přesný význam úrovní upovídanosti je na vás.

3. Kopírování

Motivace: je-li cílový adresář v jiném oddíle nebo nemá-li uživatel právo mazat soubory ze zdrojového adresáře, obyčejné přesunutí souboru selže.

Pokud přesun selže, zkusí provést kopii a smazání. Selže-li smazání, kopie v cíli zůstane. Na novém souboru nechť je stejný čas změny, jako má původní soubor.

Volitelně můžete přidat příznak, který kopírování vynutí v každém případě.

4. Úklidový režim

- `-c`, `--cleanup` – odstraní adresáře, které po přesunu neobsahují žádné soubory (ani v podadresářích).

Adresáře, které už prázdné byly (a tedy se z nich nic nepřesouvalo), mazat můžete, ale nemusíte.

5. Podpora EXIF

Neřaďte soubory podle data změny, ale podle metadat EXIF. Vizte např. balík `hsexif`. U obrázků bez metadat můžete jako fall-back použít čas změny, nebo je ignorovat. Nejlépe by chování v takovém případě mělo být konfigurovatelné. Nezapomeňte do komentáře napsat, který balík na práci s metadaty jste použili.

6. Přírůstkový režim

Přečíslujte soubory, které v cílových adresářích již jsou, aby spolu s novými tvořily souvislou posloupnost seřazenou opět podle data. Může to být výchozím chováním, nebo režimem povoleným přepínačem na příkazové řádce. „Starousedlíkům“ se prefix musí skutečně změnit, nikoli jen přidat další.

Ošetřování chyb

Protože reálný svět je kruté a nehostinné místo, musíte při práci s IO počítat s chybami takřka na každém kroku. Typicky se bude jednat o nedostatečná oprávnění pro prohledání adresáře nebo přesun souborů.

Funkce pro práci se soubory v takových případech vrhají výjimky, s nimiž se musíte rozumně popasovat. V této úloze znamená „rozumně“ většinou to, že program chybu ohlásí uživateli (který výstup je na to vhodný?) a pokračuje dál v práci. Rozhodně není žádoucí, aby kvůli nepřístupnosti jednoho adresáře selhalo celé přesouvání. Také buďte připraveni, že přesun některého souboru může selhat – např. už může existovat adresář s daným názvem.

Program by taky měl odmítnout přepsat existující soubor (což musíte ohlídat sami, knihovní funkce s tím problém nemají).

Na druhou stranu nemusíte příliš hrotit všemožné hrozící souběhy. Samozřejmě, že mezi nalezením souboru a pokusem o přečtení jeho časové známky *může* soubor přestat existovat, ale tvařme se, že se to neděje.

Tip: v základní části stačí použít `try` nebo `catch` všehovšudy jednou. Vzor „zkus akci, pokud selže, vypiš chybu, vrať nějakou výchozí hodnotu a pokračuj“ se totiž používá na více místech podobně, tudíž je vhodné jej vyčlenit do pomocného „ošetřovátka“.

Příklad

Původní obsah adresáře `Pictures`

```
Pictures
├── camera
│   ├── 2019-06-15.jpg
│   ├── 2020-03-04.jpg
│   ├── 2020-03-05.jpg
│   ├── DSC_0454.jpg
│   └── DSC_12
│       ├── DSC_1246.jpg
│       ├── DSC_1249.jpg
│       ├── DSC_1256.jpg
│       └── desc.txt
├── inf237-set08-screenshot.jpg
├── mobile
│   ├── 200105-2006.jpg
│   ├── 200105-2007.jpg
│   ├── 200229-2142.jpg
│   ├── 200229-2143.mp4
│   ├── 200301-0608.jpg
│   └── 200301-1906.jpg
├── old
│   ├── DCS_20180207.jpg
│   └── old2
│       └── DCS_20190301.jpg
└── tidy.conf
```

Obsah souboru `Pictures/tidy.conf`:

```
2020-01-03 08:24 2020-04-03 19:35 Norsko
2020-04-04 03:20 2020-04-11 18:15 karantena
```

Data vytvoření souborů jsou v příkladu stejná jako jich určuje název. Fotografie z adresáře `camera` (s názvem začínajícím na `DSC`) jsou vytvořeny dne 13. 01. 2020 v pořadí, v němž jsou vypsané. Soubor `inf237-set08-screenshot.jpg` vznikl dne 07. 04. 2020.

Po spuštění `./tidy Pictures sorted` jsou obsahy adresářů:

```
sorted
├── Norsko
│   ├── 00_200105-2006.jpg
│   ├── 01_200105-2007.jpg
│   ├── 02_DSC_0454.jpg
│   ├── 03_DSC_1246.jpg
│   ├── 04_DSC_1249.jpg
│   ├── 05_DSC_1256.jpg
│   ├── 06_200229-2142.jpg
│   ├── 07_200301-0608.jpg
│   ├── 08_200301-1906.jpg
│   ├── 09_2020-03-04.jpg
│   └── 10_2020-03-05.jpg
├── karantena
│   └── 0_inf237-set08-screenshot.jpg
└── other
    ├── 0_DCS_20180207.jpg
    ├── 1_DCS_20190301.jpg
    └── 2_2019-06-15.jpg

Pictures
├── camera
│   └── DSC_12
│       └── desc.txt
├── mobile
│   └── 200229-2143.mp4
├── old
│   └── old2
└── tidy.conf
```

Pokud by byl přítomen přepínač `-c`, adresář `old` by byl odstraněn.

Dekomposice

Úloha je poměrně dobře zvládnutelná, navrhnete-li před samotným programováním vhodnout dekomposici problému. Doporučujeme si proto po přečtení zadání nejdříve rozmyslet, z jakých podúloh utilitka sestává a rovnou si napsat typové signatury některých pomocných funkcí či typové aliasy a datové struktury. Při návrhu dekomposice může být výhodné brát ohled i na rozšíření, která budete chtít implementovat.

Vhodnost dekomposice můžete diskutovat na fóru či Discordu.

Příklady otázek k zamyšlení: *Jakou datovou strukturu použít pro mapování intervalu na řetězce, aby se v ní dobře hledalo podle jednoho bodu? Jak zajistit, aby se stejně pojmenované soubory z různých adresářů ale stejné události navzájem nepřepsaly? Kde všude budu potřebovat číst konfiguraci (byť ji zatím nemám zcela navrženou)?*

Testování

Bohužel není aktuálně v našich možnostech testovat programy s tak rozsáhlým používáním `IO`. Proto tentokrát nejsou při odevzdání spouštěny žádné automatické testy. V interaktivní osnově naleznete skript, který vyrobí testovací adresář, na němž můžete svůj program zkoušet. Testovací adresář obsahuje i několik

zajímavých případů, jako jsou omezená práva k adresáři či stejně pojmenované soubory patřící do stejné události.

Doporučujeme tento poskytnutý skript dále rozšířit a třeba si napsat i kontrolní skript, který spustí utilitku nad testovacími daty a zkontroluje, zda výsledek odpovídá očekávání.

Nemáme nic proti sdílení takovýchto testů, ať už soukromě nebo třeba přes poskytovnu. Samozřejmě ale nesmíte spolupracovat na samotném kódu řešení.

Odevzdávání

- Svou implementaci odevzdávejte do příslušné odevzdávárny v ISu. Odevzdávejte jediný soubor s příponou `.hs`, který musí být přeložitelný GHC 8.6. Zároveň platí, že všechny globálně definované funkce musí mít typovou signaturu.
- Po ukončení odevzdávání budou zveřejněna řešení všech studentů a opět dostanete na peer review dvě řešení spolustudentů.
- Peer review se zadávají přes ISovou agendu. Po rozdělení recenzentů obdržíte e-mail s pokyny.

Hodnocení

Za úlohu můžete získat až čtyři body. Pro úspěšné absolvování předmětu potřebujete alespoň jeden bod (vizte IS pro podrobné informace). Doporučujeme tedy řešení úlohy zbytečně neodkládat.

V tomto semináři nerozlišujeme body za krásu a body za funkčnost. Řešení, která budou funkční, ale ošklivá, si plný počet nezaslouží. V žádném případě byste se neměli dopouštět prohřešků, které jsou popsány zde.

Body vám udělí cvičící i se zpětnou vazbou až poté, co budou odevzdána i peer review.

Řešení tentokrát budou veřejná, takže se na ně po termínu odevzdání bude moci kdokoliv podívat. To proto, aby bylo jednodušší psát si vzájemně peer review a také proto, že by to alespoň některé mohlo motivovat k psaní hezčího kódu. Zároveň připomínáme, že už znáte nástroj HLint, který určitě není na škodu použít před odevzdáním vašeho řešení (váš opravující to určitě udělá), a přepínač `-Wall`, který by k vašemu kódu neměl mít připomínky.

Poznámky a tipy

- **Začněte pracovat včas.** Úloha není sama o sobě těžká, ale jisté množství času zabere hledání v dokumentaci a ruční testování.
- Využívat můžete libovolných modulů ze všech doposud probraných balíků; zejména `base`, `containers`, `directory` a `filepath`. Můžete použít i jiné; pak je uveďte v komentáři na začátku souboru.
- Na práci se souborovým systémem se hodit modul `System.Directory`.
- Parsování data a času obsahuje balík `time`, který už máte. Podívejte se na funkci `parseTimeM`.
- Není-li zřejmé, co se má v nějakém případě stát, udělejte to *rozumně*.
- Nejste-li si jistí, zda je něco *rozumné*, klidně se ptejte.
- Neduplikujte kód! Využívejte vlastních i knihovních funkcí.
- Nevymýšlejte znovu kolo a používejte Hoogle.
- U všech globálních funkcí uvádějte typové signatury.
- Tentokrát můžete odevzdávat ve formě modulu.
- Pokud vám není něco jasné, zeptejte se v diskusním fóru.
- V případě, že přebíráte kód odjinud, uveďte zdroj.
- Nezapomeňte, že **opisování je zakázáno** a bude postihováno podle disciplinárního řádu.