Profile manager
Vývojářská dokumentace

Jan Škoda

2. dubna 2014

Aplikace je implementovaná v jazyce C++ pro verzi standardu 11 a využívá knihovny Boost, Google logging library, xrandr, x11, gtk3, inotifytools a usb-1.0.

Organizace kódu

Aplikace sestává z několika dále uvedených hlavních funkčních celků implementovaných v C++, interních modulů zajišťujících komunikaci se systémem skrz C++ knihovny a několika externích modulů zajišťujících totéž pomocí bashových skriptů. Každý modul je potomkem třídy *Module*.

Přehled hlavních tříd podle funkčního celku:

- Parsování konfigurace – třídy *Configuration* a *Cfg_item* s potomstvem
- Parsování infixových termů lokace – třída *Term_evaluation*
- Moduly, jejich sledování a spouštění – třídy *Module_manager*, *Module* pro interní a *External_module* pro externí shellové moduly.
- Konkrétní implementace interních modulů – třídy *Module_usb* a *Module_monitor*.

Parsování konfigurace

Metoda `main()` vytvoří instanci třídy Configuration, která naparsuje konfiguraci do statických `std::map` *senses,actions,locations* a *rules*. K tomu využívá šablonové funkce `add_action(mapa, polozka)`, která přidá položku libovolné podtřídy `Cfg_item` do příslušné mapy. Pokud parsování narazí na hlavičku sekce (řádku ve tvaru `[nazev-sekce]`), nastaví do proměnné `section_method` částečně aplikované volání (vizte *Curryfikace*,`std::bind`) metody `add_action`. Parsování konfigurace se pak tedy skládá jen z vytváření instancí podtříd `Cfg_item` a volání metody `add_action`, která tuto instanci přidá do příslušné mapy.

Parsování termů v definicích lokací

Lokace jsou v konfiguraci zadány logickým termem složeným z identifikátorů senzorů a logických operací. Ten je třeba z infixové notace z důvodu efektivity předzpracovat do postfixové. Do postfixové notace se pak dosazuje po každé změně hodnoty některého ze senzorů aby se ověřila přítomnost v lokaci. Pro převod do postfixové notace je použit algoritmus využívající frontu a zásobník znám též jako *Shunting-yard algoritmus*.

Logika a implementace modulů

Moduly jsou celky, poskytující programu interakci se systémem. A to jak vstup (senses), tak výstup (actions). Jeden modul může poskytovat oba směry interakce. Každý modul implementuje virtuální metody `string get_status(const string& variable)` a `void set(const string& variable, const string& value)`. První z nich získá čárkou oddělený seznam hodnot, kterým je zadaná *variable* rovna.

Tato metoda implementuje sense/smysl, vstupní funkci modulu. Druhá nastaví proměnnou na zadanou hodnotu. Tato metoda implementuje akci, výstup.

Každý modul též implementuje virtuální metodu `void watch()`, která vytvoří jeho vlákno obsluhující změny v systému. Všechny interní moduly z důvodů efektivity využívají vždy jen pasivní čekání (poll, inotify). Pokud dojde ke změně, vyžádá si modul nové vyhodnocení přítomnosti v lokacích (dosazení do termů) zavoláním statické metody `void reprocess()`. Tato metoda je thread-safe díky mutexovému lock_guardu.

Externí moduly realizují zmíněné vstupně výstupní metody voláním příslušného bashového skriptu. Jejich účelem je umožnit uživateli, aby snadno mohl přidávat vlastní funkcionalitu, kterou bude Profile manager umět automaticky využívat. Metoda `watch` je u smyslových externích modulů implementována aktivním čekáním, tj. každých 5 sekunds znovu spouští skript a dotazuje se jej na změny.

Rozhraní externích modulů

Každý externí modul je bashový skript ve složce `ext-modules`. Při spuštění skriptu bez parametrů vypíše skript na výstup na každé řádce jednu proměnnou a její hodnotu oddělené rovnítkem, kolem nějž mohou být mezery. Pokud je skript zavolán s dvěma parametry, provede v systému změnu dle svého účelu. Prvním parametrem je identifikátor proměnné a druhým její hodnota. Pokud změna uspěje, musí vypsat jedinou řádku `SET`. Například modul *xrandr* při spuštění s parametry *vga-resolution* a *1024x768* nastaví rozlišení VGA zařízení.