Dokumentace úlohy CHA: C Header Analysis v Python 3 do IPP 2014/2015
Jméno a příjmení: Jakub Pastuszek
Login: xpastu00

Dokumentace k projektu IPP 2014/2015, CHA: C Header Analysis

Analýza zadání

Cílem projektu bylo vytvořit skript v jazyce Python 3, který bude provádět analýzu hlavičkových souborů jazyka C dle standardu ISO C99 a bude spustitelný na serveru Merlin. Skript vytvoří databázi nalezených funkcí. Skript lze spouštět s různými přepínači, a tím upravovat podobu výstupu.

Postup řešení

Prvotní událostí při spuštění skriptu je definice vstupních parametrů, které budou převzaty jako přepínače skriptu a jejich následná kontrola. Dále definice třídy pro obsluhu chybových stavů. Poté jsou definovány proměnné pro udržení stavu automatu. Následuje funkce, která řeší vymazání všech maker, komentářů a řetězců ze zdrojového souboru pomocí průchodu konečným automatem. Obsahem každého nechybového výstupu je xml hlavička a element *functions*.

Nejdůležitější částí celého skriptu je funkce s názvem *parseFile*, která zpracovává vstupní soubor a hledá v něm výskyty definic funkcí. Tato analýza a následné nalezení definic se provádí pomocí regulárního výrazu. Každá nalezená definice se zpracovává odděleně, nezávisle na ostatních (pokud nejsou uvedeny určité přepínače, viz níže). Zpracovávaná funkce je rozdělena na návratový typ, jméno a parametry. Parametry jsou ještě dále analyzovány pro zjištění jednotlivých argumentů.

Dále je třeba nastavit výstupní soubor, který může být zadán přepínačem *--output*, jinak se vypisuje na standardní výstup. Také je nutné zjistit, zda zadaný vstupní soubor je adresář, soubor, nebo neexistující soubor.

Pro průchod adresářem jsem použil metodu os.walk(), která je prováděna ve 3 zanořených cyklech.

Funkce

Pomocná funkce s názvem *automata* (schéma automatu viz Obr. 1) prochází vstupní soubor znak po znaku a zjišťuje, zda neobsahuje makra, komentáře, či řetězce (dále jen „věci navíc“). Tento proces je prováděn pomocí průchodu konečným automatem, kdy vše, co není věci navíc, se zapíše to výstupního řetězce a řádky/bloky kódu, které jsou nebo by mohly být věcmi navíc jsou zpracovány automatem. Např. lomeno může značit začátek komentáře, nebo dělení, ale to lze zjisti až podle následujícího znaku. Dále taky komentář může být buď řádkový, nebo blokový.

Přepínač *--pretty-xml* určuje, jak má být výstupní soubor formátován. Když je zadán tento přepínač bez hodnoty, je použito odřádkování každého elementu a odsazení o 4 mezery, při každém dalším zanoření se počet mezer zdvojnásobuje. Může být také explicitně uvedena hodnota (číslo) o kolik mezer se má každé zanoření odsazovat, nejmenší odsazení lze použít 0, kdy výstup bude po každém elementu pouze odřádkován. Odstranění přebytečných bílých znaků ve výpisu jednotlivých atributů, které jsou následně nahrazeny jednou mezerou, je možné při použití přepínače *--remove-whitespace*. Všechny tyto přepínače ovlivňují chování funkce *parseFile* a formátování výstupu.

Ostatní přepínače jen zakazují výpis některých funkcí dle určitých kritérií. Např. *--no-inline* zakáže vypsání funkce která je inline, *--max-par=n* nebude vypisovat funkce, které mají více než *n* parametrů. Přepínač *--no-duplicates* nevypíše duplicitní deklarace funkce. Tyto přepínače ovlivňují pouze chování funkce *parseFile*.

Zpracování parametrů

Parametry nejsou zpracovávány pomocí žádné vestavěné funkce. Funkce, která zpracovává parametry skriptu, úzce spolupracuje s třídou *ArgERR*, které se stará o výpis chybové hlášky a uvedení správného návratového kódu. Vyhledávání jednotlivých přepínačů probíhá v poli řetězců *argv*, kde každý řetězec je zpracováván zvlášť. Pokud je nalezen neplatný přepínač nebo je některý přepínač předefinován (je uveden více než 1x), je skript ihned ukončen s návratovým kódem 1.

Zpracování vstupních souborů

Vstupní soubor je po zpracování automatem načten jako souvislé pole znaků (řetězec), v Python 3 reprezentováno datovým typem *string*. Pokud je zadán vstupní soubor, zpracovává se bez ohledu na koncovku, v případě zadání adresáře se zpracovávají všechny soubory s koncovkou .h, a dále soubory (také s koncovkou .h) ve všech adresářích při rekurzivním průchodu podadresářů.

Závěr

Projekt jsem z počátku řešil na svém PC, kde jsem měl nainstalovaný Python 3.4.2 server, na kterém jsem si zkoušel funkčnost skriptu. Před pokusným odevzdáním jsem začal testovat na školním serveru Merlin. Testování regulárních výrazů jsem prováděl na serveru www.regex101.com.

S jazykem Python 3 jsem se dříve nesetkal, proto pro mě bylo výzvou se tento jazyk, alespoň zlehka naučit. Tento jazyk je dle mého názoru velice mocný. Na téměř každou operaci má nějakou funkci nebo krátký blok kódu, který požadovaný problém řeší. S jazykem jako takovým jsem neměl žádný problém, po nainstalování doplňku pro Python do Microsoft Visual Studia v12.0 jsem ani se zajímavým řešení bloků v Pythonu neměl sebemenší problém. V tomto jazyce se dá skvěle pracovat s regulárními výrazy, proto jsem neváhal a také jsem je ve skriptu použil.

Příloha

Obr. 1: Schéma konečného automatu