ASSIGNMENT # 1

CPSC 411 Winter 2001 Lec ?? Lab 02 Student: Chad C. D. Clark < clarkch @ cpsc . ucalgary . ca > Date: Jan 28, 2002 Synopsis: A minusculus compiler written in pure C that generates stack machine code as defined in file AS1.SPECS. NOTE: I did use the _OLD_ grammar. Though the rules do get manipulated a bit. Refer to source code comments for term, expr and, stmtlist. Input: Reads minusculus source code from STDIN. Output: Writes stack machine target code to STDOUT. Errors (including warnings) are reported to STDERR. Files: README - contains info about this project and tarball. AS1.SPECS - copy of the assignment in text format. AS1.GRAMMAR - new grammar (non left-recursive). asllex.1 - the lexer and main() routine. aslparse.c - performs the parsing and syntax tree construction. asltree.c - function that deal with building the tree and traversing. as1globals.h - info used by more than one file. asltokens.h - defines the token types used by the lexer and parser. asltree.h - defines node types and tree functions. makefile - pretty standard makefile. tests/ - directory with many test files. tests/do-tests - script to run the tests found in tests directory. Compiling: Just use the command "make". "make clean" deletes lex.yy.c, *.o and a.out "make stats" counts source file lines. Requires flex and gcc - Developed with: - flex 2.5.4 - egcs 2.91.66 - gdb 4.18 Postmortum: (Mostly for my benifit but I wanted to keep this next to the code.) Things that went well: - Writing the tree pointer code. Only one null pointer error! It was due to not checking for NULL and not having an END_NODE. - Writing the lexer. LEX rocks! Things that went not so well: - Getting the shape of the tree stucture down in my mind. I was starting to get two disjoint models (parse code and node building code) then I sat down, drew pictures and, traced out tree traversals untill I had a model. Then writing down a record of the model for reference. - Realizing C unlike C++ requires 'struct' before every declaration. Things that were usefull: - Drawing pictures (trees in particular). - Writing down plans, decisions and models for reference in other parts of the program. - Writing code on paper before typing it up. - GDB and the GDB book by Stallman and Pesch.

__README__.txt

- syntax highlighting & 60 row console mode

Things I would like to do differently on future projects:

- Sit down and plan out the structure of the program before writing code.
- Document my data structures before using them. It's easier than searching with grep and opening up an editor.

Things I learned:

- A nice (ie non left recursive grammar) is very usefull.
- Follow sets can make parsing much easier.

Things I want to learn: - How to let flex let me define main() outside of the .l file.