Lab 3 Makefiles

Learning Outcomes

- Observe the action of 'git clone'
- Explain the purpose for a makefile
- Define the terms target and dependency
- Compare and contrast implicit and explicit rules
- Construct a simple makefile

GIT Intro and GIT Clone

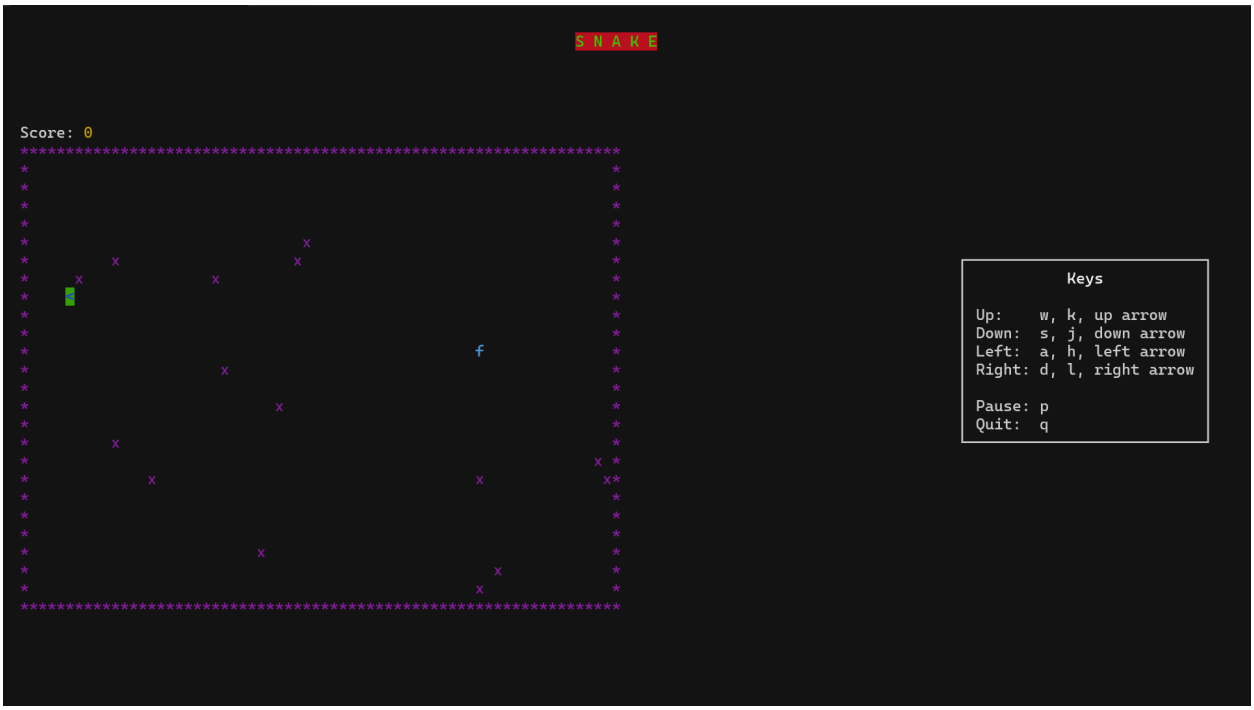

Figure 1: terminal result of compiled program

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ gcc *.c -o snakes-game -l ncurses
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ ls
README.md  arguments_parser.c  config.h  field.h  score_file.c  snake.c  snakes-game
a.out      arguments_parser.h  field.c   game.c   score_file.h  snake.h
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ ./snakes-game
Player 1: New high score (0 points)
Your name [(null)]: Leigh Goetsch

                    TOP SCORES
-------------------------------------------------
| Top | Score | Player Name   | Date             |
-------------------------------------------------
|   1 | 0     | null          | 17-09-2024 15:24 |
|   2 | 0     | Leigh Goetsch | 17-09-2024 15:25 |
-------------------------------------------------
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$
```

Figure 2: Command to compile the project and result of program running

[1]

[1] include a screenshot of the successful build (showing command used to build on the console)

Dependecies

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ gcc snake.c -M
snake.o: snake.c /usr/include/stdc-predef.h snake.h field.h \
 /usr/include/stdlib.h \
 /usr/include/x86_64-linux-gnu/bits/libc-header-start.h \
 /usr/include/features.h /usr/include/features-time64.h \
 /usr/include/x86_64-linux-gnu/bits/wordsize.h \
 /usr/include/x86_64-linux-gnu/bits/timesize.h \
 /usr/include/x86_64-linux-gnu/sys/cdefs.h \
 /usr/include/x86_64-linux-gnu/bits/long-double.h \
 /usr/include/x86_64-linux-gnu/gnu/stubs.h \
 /usr/include/x86_64-linux-gnu/gnu/stubs-64.h \
 /usr/lib/gcc/x86_64-linux-gnu/11/include/stddef.h \
 /usr/include/x86_64-linux-gnu/bits/waitflags.h \
 /usr/include/x86_64-linux-gnu/bits/waitstatus.h \
 /usr/include/x86_64-linux-gnu/bits/floatn.h \
 /usr/include/x86_64-linux-gnu/bits/floatn-common.h \
 /usr/include/x86_64-linux-gnu/sys/types.h \
 /usr/include/x86_64-linux-gnu/bits/types.h \
 /usr/include/x86_64-linux-gnu/bits/typesizes.h \
 /usr/include/x86_64-linux-gnu/bits/time64.h \
 /usr/include/x86_64-linux-gnu/bits/types/clock_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/clockid_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/time_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/timer_t.h \
 /usr/include/x86_64-linux-gnu/bits/stdint-intn.h /usr/include/endian.h \
 /usr/include/x86_64-linux-gnu/bits/endian.h \
 /usr/include/x86_64-linux-gnu/bits/endianness.h \
 /usr/include/x86_64-linux-gnu/bits/byteswap.h \
 /usr/include/x86_64-linux-gnu/bits/uintn-identity.h \
 /usr/include/x86_64-linux-gnu/sys/select.h \
 /usr/include/x86_64-linux-gnu/bits/select.h \
 /usr/include/x86_64-linux-gnu/bits/types/sigset_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/__sigset_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/struct_timeval.h \
 /usr/include/x86_64-linux-gnu/bits/types/struct_timespec.h \
 /usr/include/x86_64-linux-gnu/bits/pthreadtypes.h \
 /usr/include/x86_64-linux-gnu/bits/thread-shared-types.h \
 /usr/include/x86_64-linux-gnu/bits/pthreadtypes-arch.h \
 /usr/include/x86_64-linux-gnu/bits/atomic_wide_counter.h \
 /usr/include/x86_64-linux-gnu/bits/struct_mutex.h \
 /usr/include/x86_64-linux-gnu/bits/struct_rwlock.h /usr/include/alloca.h \
 /usr/include/x86_64-linux-gnu/bits/stdlib-float.h /usr/include/time.h \
 /usr/include/x86_64-linux-gnu/bits/time.h \
 /usr/include/x86_64-linux-gnu/bits/types/struct_tm.h \
 /usr/include/x86_64-linux-gnu/bits/types/struct_itimerspec.h \
 /usr/include/x86_64-linux-gnu/bits/types/locale_t.h \
 /usr/include/x86_64-linux-gnu/bits/types/__locale_t.h
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$
```

Figure 3: results of gcc snake.c -M

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ cat snake.c
/*
 * Copyright (C) 2020 Esteban López Rodríguez <gnu_stallman@protonmail.ch>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <https://www.gnu.org/licenses/>.
 */

// modification of header location for CPE2600 - DER 9/10/2023
#include "snake.h"
#include <time.h>

snake_t*
init_snake(field_t *field, cell_t head_type)
{
        snake_t *snake;

        snake = malloc(sizeof(snake_t));
```

Figure 4: beginning of the snake.c file, showing only snake.h and time.h included

[2]

The header files listed with the -M command include dependencies of the files included. Proof of this is that `snake.c` includes `snake.h` which includes `field.h`, which is not included by `snake.c`, but is listed. [3]

[2] Include at least one screen cap as proof.

[3] explain why so many header files are listed by gcc when only two files are included.

A Simple Makefile

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make
gcc -c -Wall snake.c -o snake.o
gcc -MM snake.c > snake.d
gcc -c -Wall arguments_parser.c -o arguments_parser.o
gcc -MM arguments_parser.c > arguments_parser.d
gcc -c -Wall field.c -o field.o
gcc -MM field.c > field.d
gcc -c -Wall game.c -o game.o
gcc -MM game.c > game.d
gcc -c -Wall score_file.c -o score_file.o
gcc -MM score_file.c > score_file.d
gcc snake.o arguments_parser.o field.o game.o score_file.o
 -l ncurses -o snake_game
```

Figure 5: result of sucessful build using make

[4]

Testing Incremental Build

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make
make: Nothing to be done for 'all'.
```

Figure 6: result of make with no changes

[5] The result of making after no changes to do nothing makes sense, since there is nothing to rebuild.[6]

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ rm field.o
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make
gcc -c -Wall field.c -o field.o
gcc -MM field.c > field.d
gcc snake.o arguments_parser.o field.o game.o score_file.o -l ncurses
-o snake_game
```

Figure 7: result of deleting field.o and calling make again

[4] grab a screen cap of the console with your first successful build
[5] Without changing any of the source files, type the command 'make' again. What happens?
[6] Does this make sense?

[7] When make is called field.c was rebuilt. [8] The result of this call makes sense since the field.c result/object file was changed.[9]

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ touch field.c
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make
gcc -c -Wall field.c -o field.o
gcc -MM field.c > field.d
gcc snake.o arguments_parser.o field.o game.o score_file.o -l ncurses
-o snake_game
```

Figure 8: result of 'editing' field.c and calling make again

[10]When make was called, field.c was rebuilt.[11] It makes sense that make will rebuild the field.c file because the file has been flagged as changed since it was last built.[12] The compiler knew to recompile field.c because it was changed. [13]

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ touch field.h
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make
gcc -c -Wall snake.c -o snake.o
gcc -MM snake.c > snake.d
gcc -c -Wall field.c -o field.o
gcc -MM field.c > field.d
gcc -c -Wall game.c -o game.o
gcc -MM game.c > game.d
gcc snake.o arguments_parser.o field.o game.o score_file.o -l ncurses
 -o snake_game
```

Figure 9: result of 'editing' field.h and calling make again

[14] When make is called, snake.c, field.c, and game.c were rebuilt.[15] This makes sense because all three c files include field.h. [16]

[7] Delete field.o (rm field.o). Run 'make'.
[8] What happens?
[9] Does this make sense?
[10] Edit field.c (or just update its timestamp with 'touch field.c') and Run 'make'.
[11] What happens?
[12] Does this make sense?
[13] How did make know to recompile field.c?
[14] Edit field.h (or just update its timestamp with 'touch field.h') and Run 'make'.
[15] What happens?
[16] Does this make sense? Be very specific with this answer.

```
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ make clean
rm -rf snake.o arguments_parser.o field.o game.o score_file.o snake_ga
me
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$ ls
Makefile            config.h  game.d        snake.d
README.md           field.c   score_file.c  snake.h
arguments_parser.c  field.d   score_file.d
arguments_parser.d  field.h   score_file.h
arguments_parser.h  game.c    snake.c
goetschm@AAD-PF50KM51:~/cpe2600/lab3/snake$
```

Figure 10: result of calling make clean

Calling make clean removes the .o files and the executable after building. [17]

[17] Issue command 'make clean'. What happens?