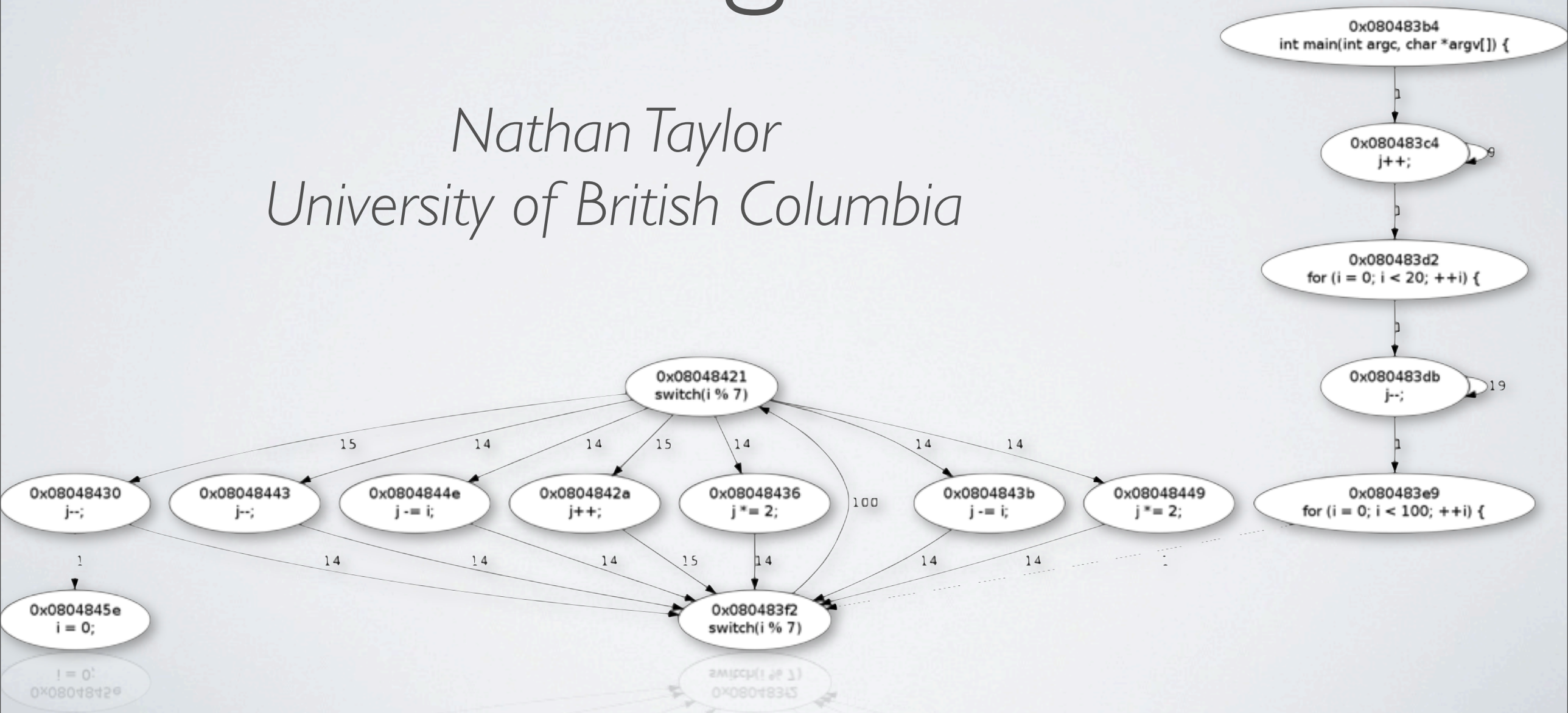
Event Loops as Execution Signatures

Nathan Taylor
University of British Columbia



```
  while (TRUE)
    {
      XEvent xevent;
      XNextEvent (xdisplay, &xevent);
      sn_display_process_event (display, &xevent);
    }
```

```
MSG msg;
while(GetMessage(&msg, hwnd, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
 ...
}
```

```
  GetEventParameter (event, kEventParamDirectObject, typeHICommand, NULL,
                                    sizeof (HICommand), NULL, &command);
    switch (command.commandID)
      {
   ...
      }
    return result;
```

XLib

```
  while (TRUE)
    {
      XEvent xevent;
      XNextEvent (xdisplay, &xevent);
      sn_display_process_event (display, &xevent);
    }
```

Win32

```
MSG msg;
while(GetMessage(&msg, hwnd, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
 ...
}
```

Carbon

```
  GetEventParameter (event, kEventParamDirectObject, typeHICommand, NULL,
                                    sizeof (HICommand), NULL, &command);
    switch (command.commandID)
      {
   ...
      }
    return result;
```

```
for (; ;) {
  connfd = Accept(listenfd, NULL, NULL)
          ...
  Write(connfd, buff, strlen(buff);
  Close(buff);
}
```

```
 607   /* The Main Loop.  The Big Cheese.  The Top Dog.  The Head Honcho.  The.. */
 608   while (!circle_shutdown) {
           ...
 680     if (select(maxdesc + 1,&input_set,&output_set,&exc_set,&null_time) < 0) {
 681         perror("SYSERR: Select poll");
 682         return;
 683     }
```

```
2073     while ( 1 ) {
2074         ev = Com_GetEvent();
2075
           ...
2082
2083             while ( NET_GetLoopPacket( NS_SERVER, &evFrom, &buf ) ) {
2084                 // if the server just shut down, flush the events
2085                 if ( com_sv_running->integer ) {
2086                     Com_RunAndTimeServerPacket( &evFrom, &buf );
2087                 }
2088             }
2090             return ev.evTime;
2091         }
```

TCP Daytime server (UNP)

```
for (; ;) {
  connfd = Accept(listenfd, NULL, NULL)
          ...
  Write(connfd, buff, strlen(buff);
  Close(buff);
}
```

CircleMUD

```
 607   /* The Main Loop.  The Big Cheese.  The Top Dog.  The Head Honch
 608   while (!circle_shutdown) {
           ...
 680     if (select(maxdesc + 1,&input_set,&output_set,&exc_set,&null_time) < 0) {
 681         perror("SYSERR: Select poll");
 682         return;
 683     }
```

Quake III multiplayer server

```
2073     while ( 1 ) {
2074         ev = Com_GetEvent();
2075
           ...
2082
2083             while ( NET_GetLoopPacket( NS_SERVER, &evFrom, &buf ) ) {
2084                 // if the server just shut down, flush the events
2085                 if ( com_sv_running->integer ) {
2086                     Com_RunAndTimeServerPacket( &evFrom, &buf );
2087                 }
2088             }
2090             return ev.evTime;
2091         }
```

SPACE OF ALL PROGRAMS

...partitioned by how their main loop behaves...?

http://en.wikipedia.org/wiki/File:Set_partition.svg

EXAMPLE USECASES

- **Reverse engineering**: Provides a sketch of how a program behaves without requiring source. In malware samples, main loop might consist of unpackers, interpreters, and other fun stuff

- **System-wide analysis**: Categorizing all running processes on a system into their equivalence classes to find suspicious outliers

PROOF OF CONCEPT TOOL

PROOF OF CONCEPT TOOL

- **Superblock jump graph generation**

PROOF OF CONCEPT TOOL

- **Superblock jump graph generation**
- **Memory diffs between loop iterations**

PROOF OF CONCEPT TOOL

- **Superblock jump graph generation**
- **Memory diffs between loop iterations**

- Valgrind / Binutils parsing
- Vertex contraction / miscellaneous graph algorithms
- Generating pretty pictures (!)

FIRST GOAL

"From the behaviour of a running program, determine a reasonable starting location for what could be the program's main loops"

GUESSING CRITERIA

GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program

GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program
    - Look for vertices with large weights on in-edges

GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program
    - Look for vertices with large weights on in-edges
- 2. A desirable loop will be first entered **relatively early** on in the execution trace

GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program
    - Look for vertices with large weights on in-edges
- 2. A desirable loop will be first entered **relatively early** on in the execution trace
    - Look for vertices with a short path from **`main()`**

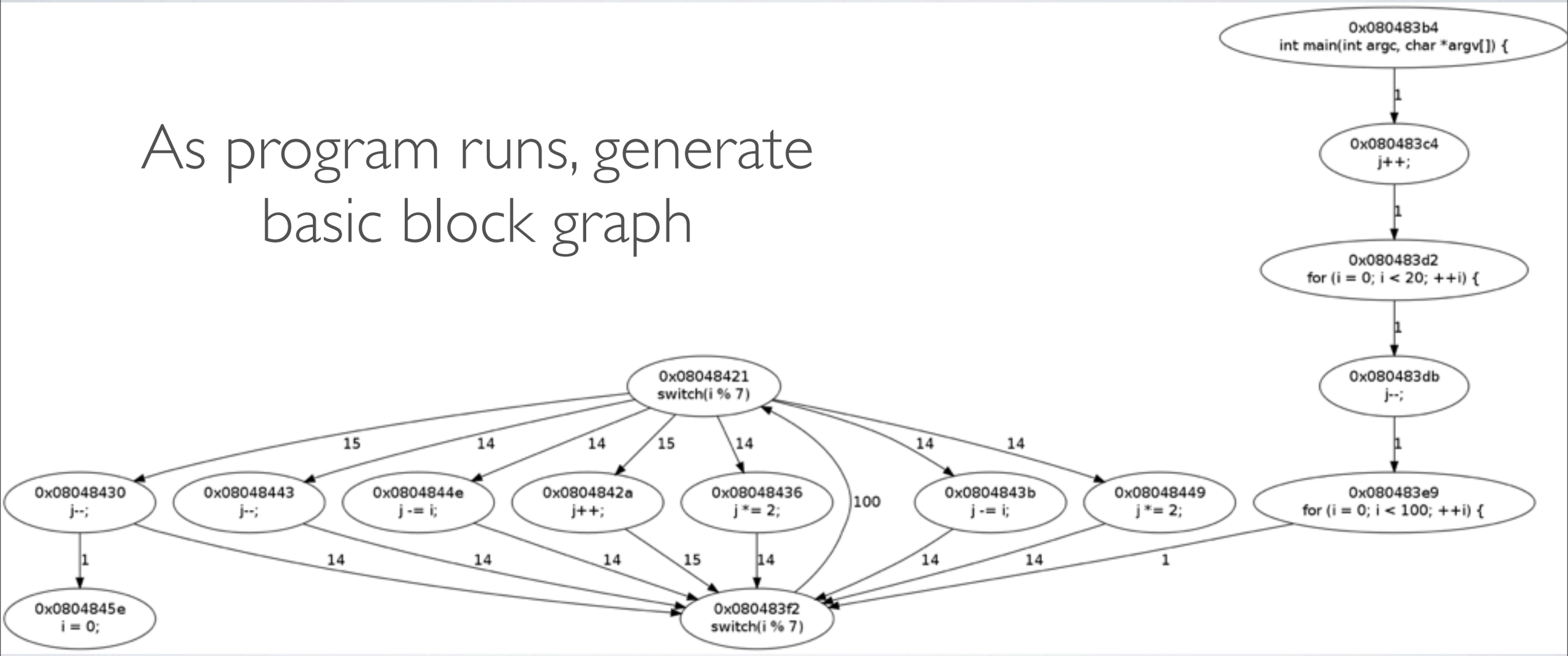
GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program
    - Look for vertices with large weights on in-edges
- 2. A desirable loop will be first entered **relatively early** on in the execution trace
    - Look for vertices with a short path from **`main()`**
- 3. A desirable loop will be executed within a **stack frame relatively close** to `main()`'s

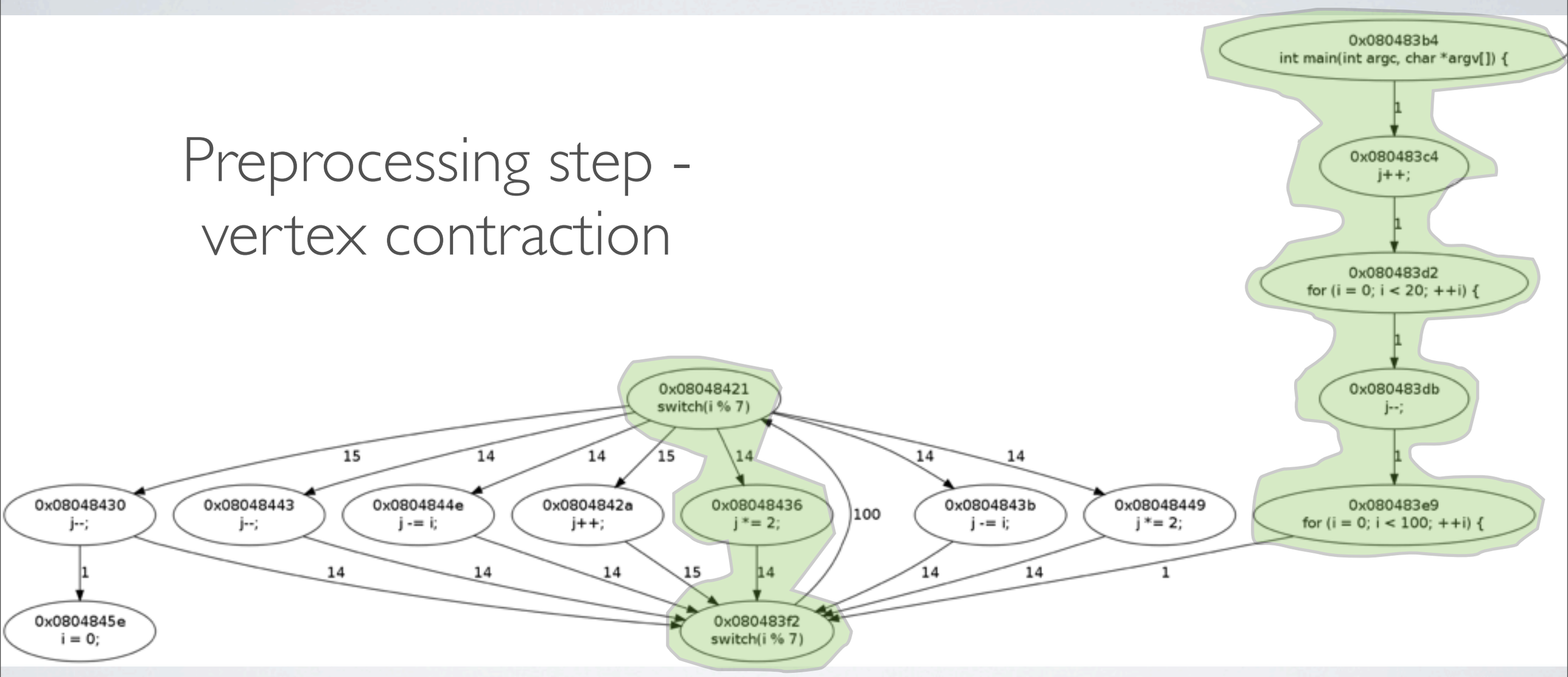
GUESSING CRITERIA

- 1. A desirable loop will be **entered more frequently** than other parts of the program
    - Look for vertices with large weights on in-edges
- 2. A desirable loop will be first entered **relatively early** on in the execution trace
    - Look for vertices with a short path from **`main()`**
- 3. A desirable loop will be executed within a **stack frame relatively close** to `main()`'s
    - Edge weight decreases exponentially as distance from `main()`'s ebp increases

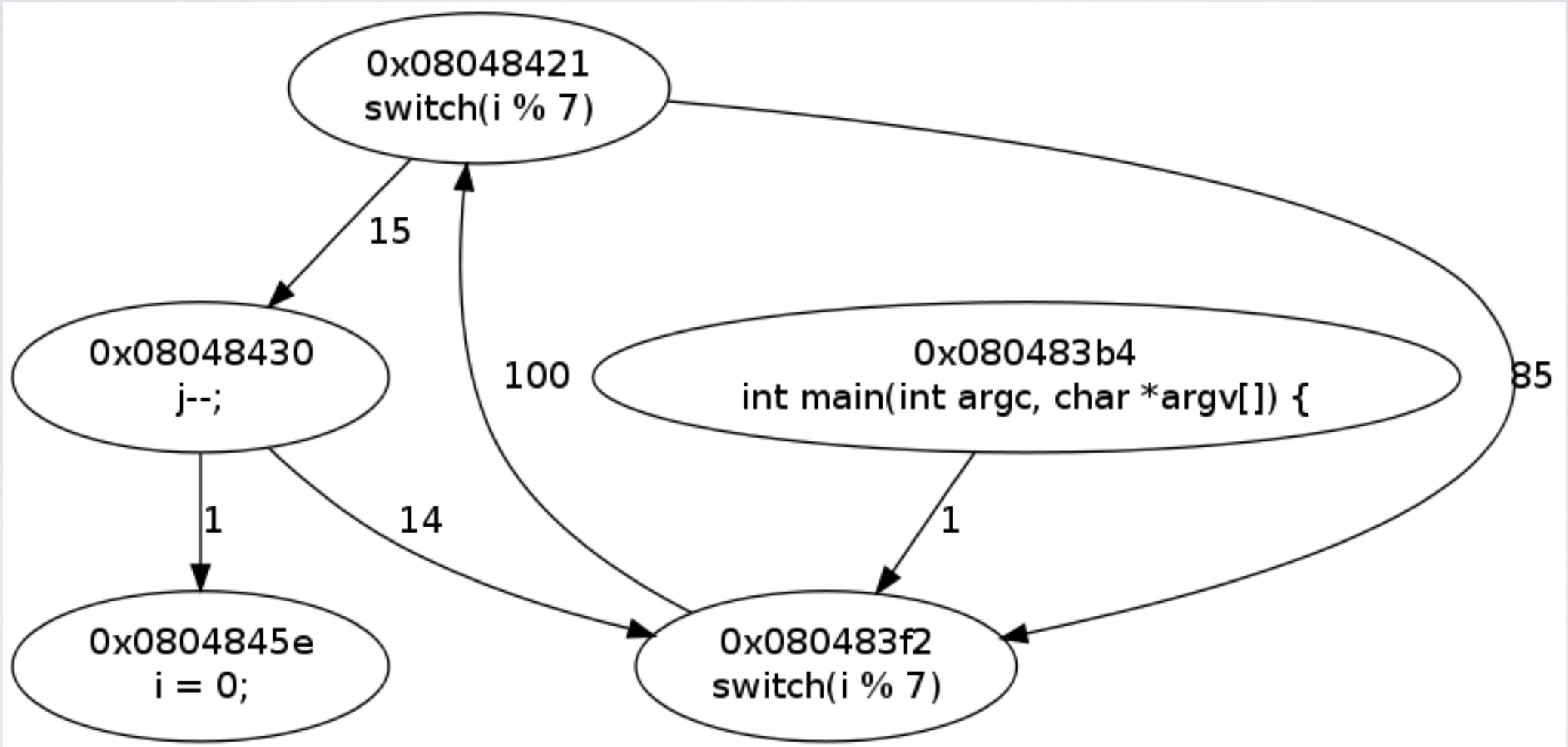
As program runs, generate basic block graph



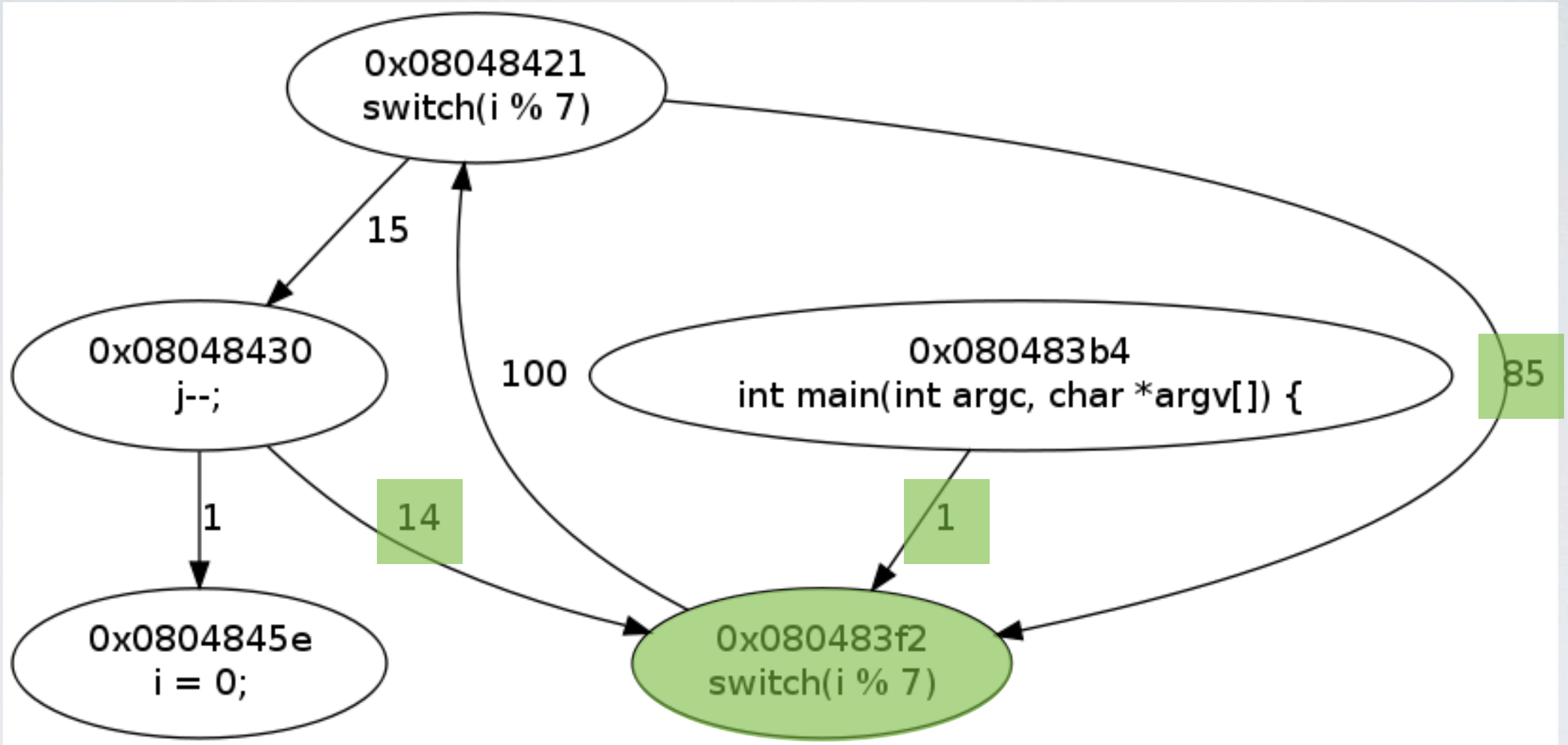
Preprocessing step - vertex contraction

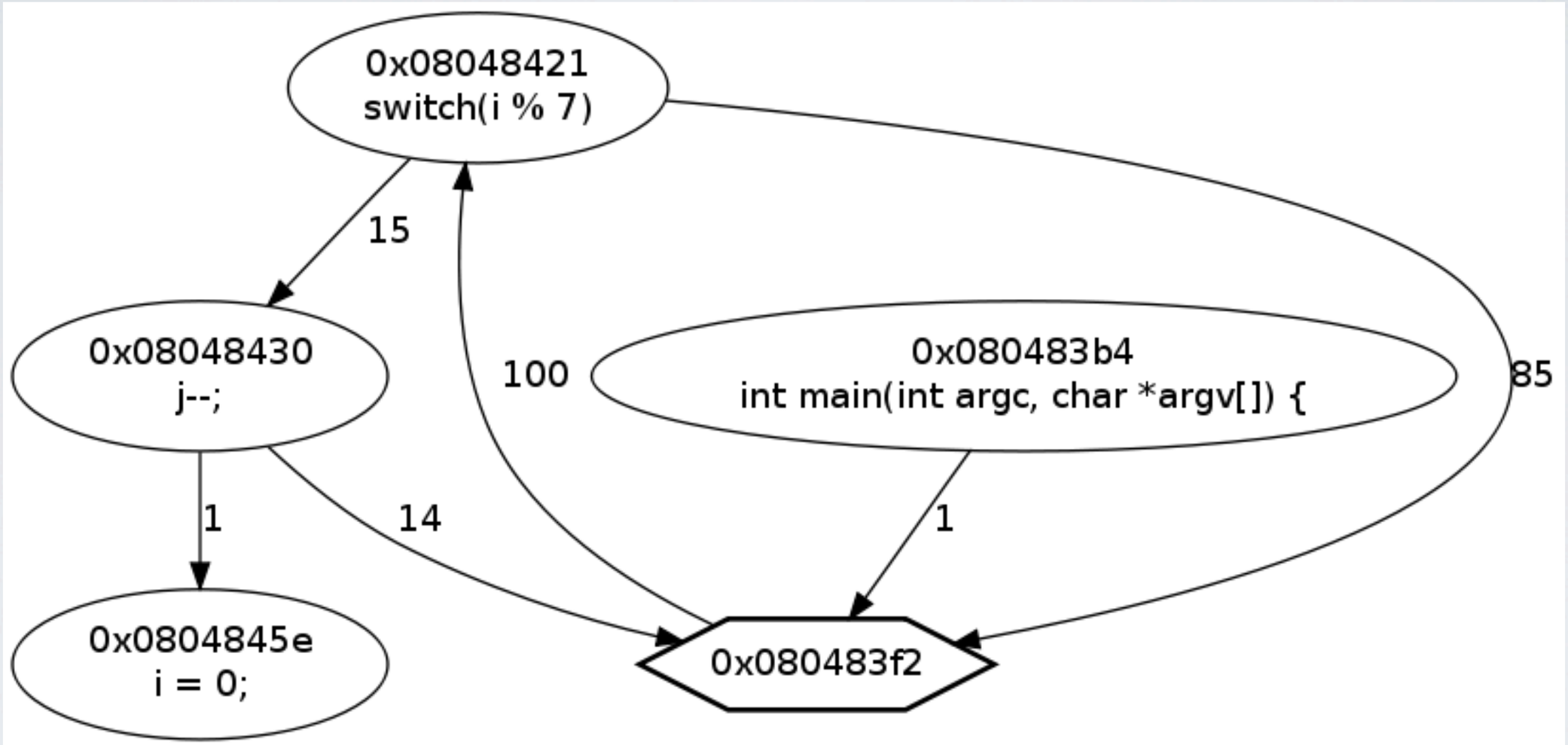


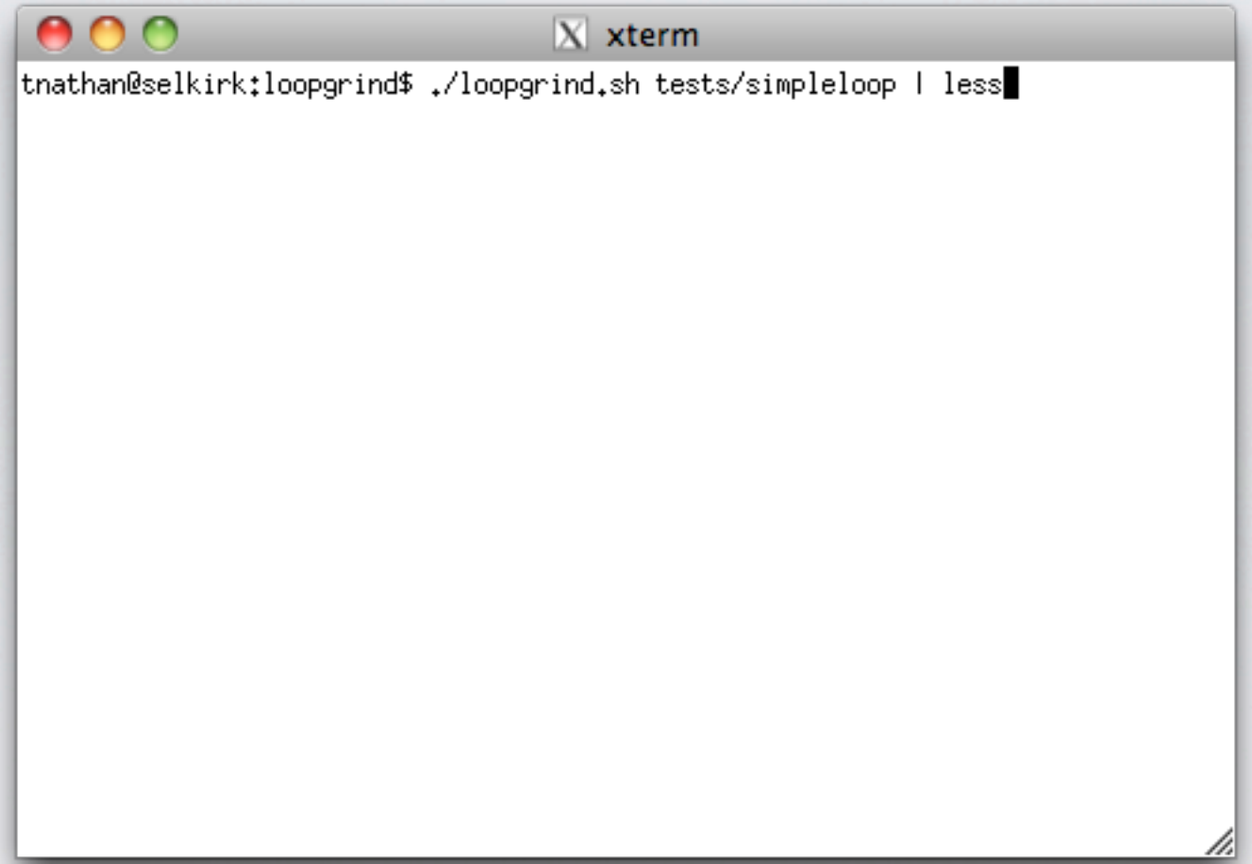
Apply criteria 1-3

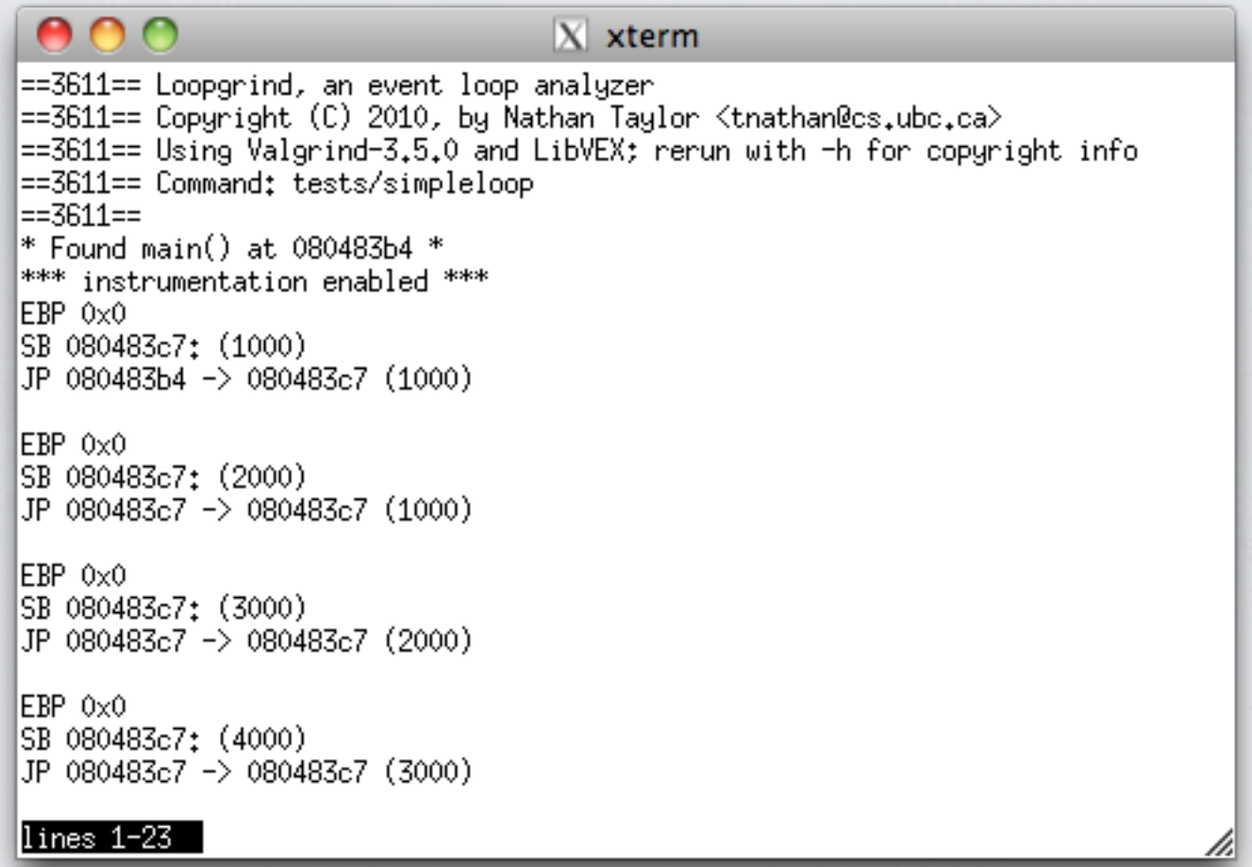


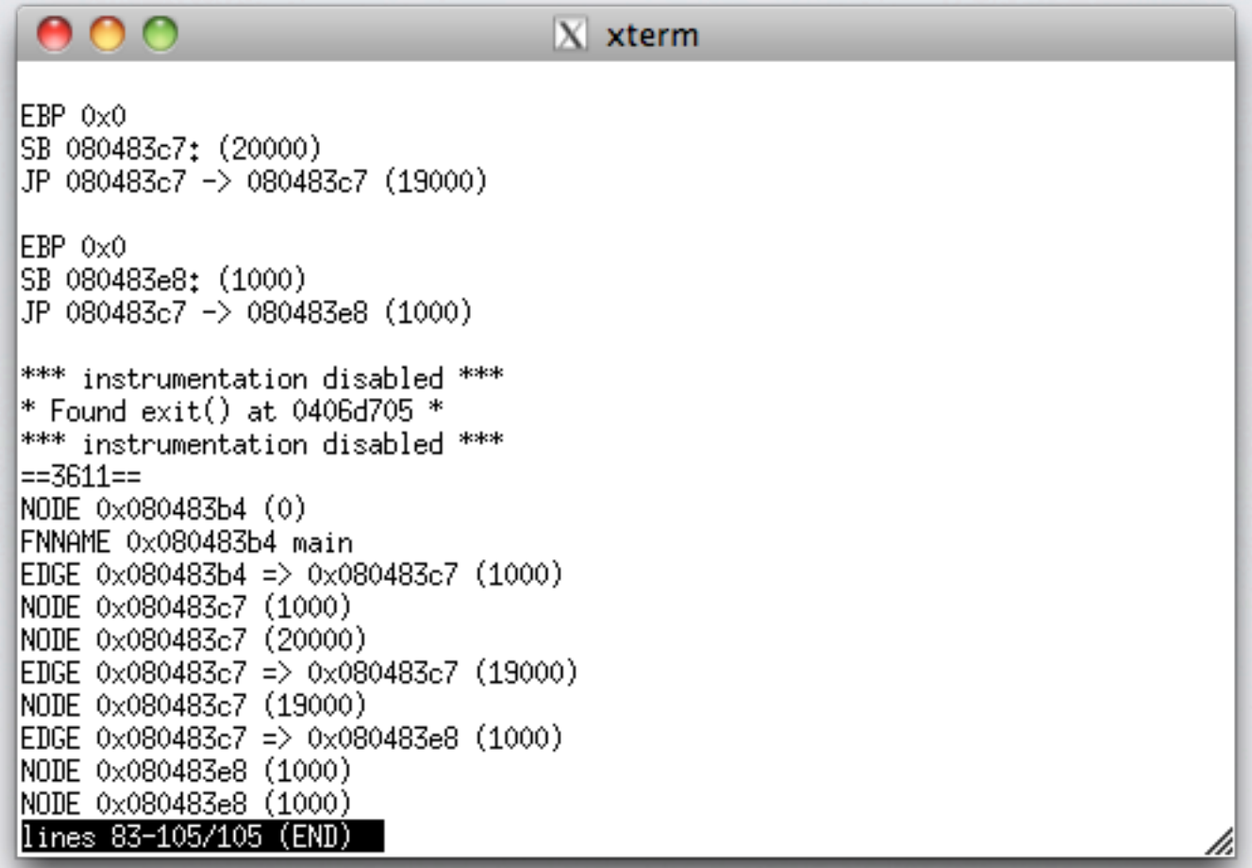
Apply criteria 1-3

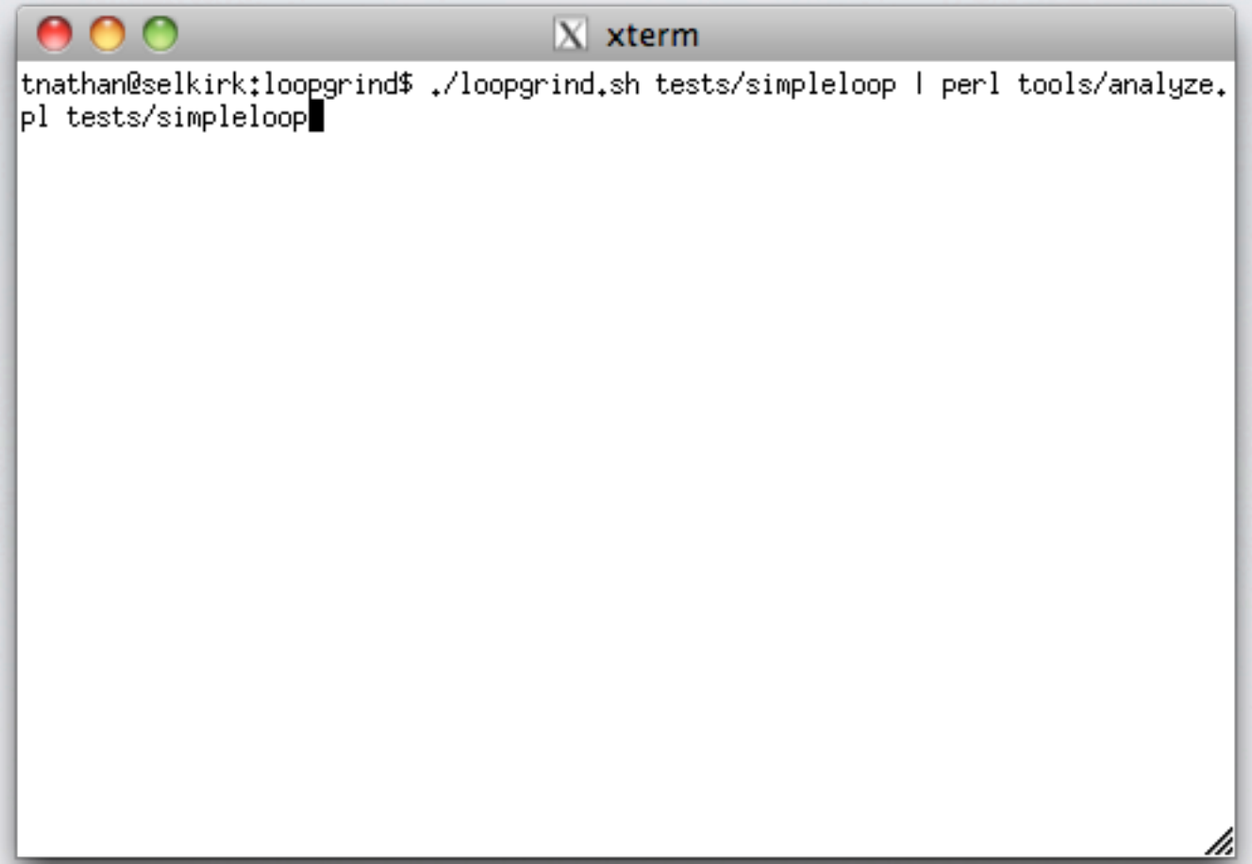


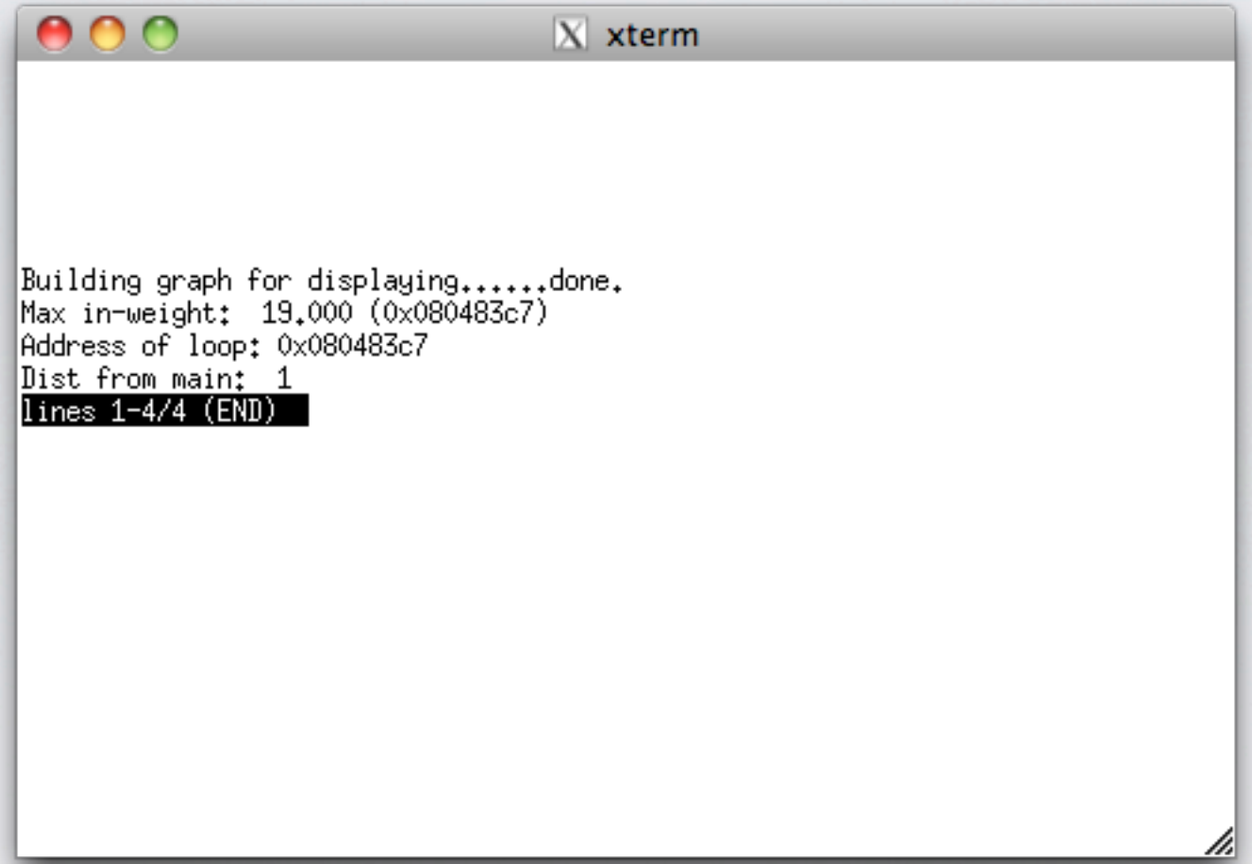

0x08048421
switch(i % 7)
15
0x08048430
j--;
100
0x080483b4
int main(int argc, char *argv[]) {
85
1
14
1
0x0804845e
i = 0;
0x080483f2


xterm
tnathan@selkirk:loopgrind$ ./loopgrind.sh tests/simpleloop | less


xterm
==3611== Loopgrind, an event loop analyzer
==3611== Copyright (C) 2010, by Nathan Taylor <tnathan@cs.ubc.ca>
==3611== Using Valgrind-3.5.0 and LibVEX; rerun with -h for copyright info
==3611== Command: tests/simpleloop
==3611==
* Found main() at 080483b4 *
*** instrumentation enabled ***
EBP 0x0
SB 080483c7: (1000)
JP 080483b4 -> 080483c7 (1000)
EBP 0x0
SB 080483c7: (2000)
JP 080483c7 -> 080483c7 (1000)
EBP 0x0
SB 080483c7: (3000)
JP 080483c7 -> 080483c7 (2000)
EBP 0x0
SB 080483c7: (4000)
JP 080483c7 -> 080483c7 (3000)
lines 1-23


xterm
EBP 0x0
SB 080483c7: (20000)
JP 080483c7 -> 080483c7 (19000)
EBP 0x0
SB 080483e8: (1000)
JP 080483c7 -> 080483e8 (1000)
*** instrumentation disabled ***
* Found exit() at 0406d705 *
*** instrumentation disabled ***
==3611==
NODE 0x080483b4 (0)
FNNAME 0x080483b4 main
EDGE 0x080483b4 => 0x080483c7 (1000)
NODE 0x080483c7 (1000)
NODE 0x080483c7 (20000)
EDGE 0x080483c7 => 0x080483c7 (19000)
NODE 0x080483c7 (19000)
EDGE 0x080483c7 => 0x080483e8 (1000)
NODE 0x080483e8 (1000)
NODE 0x080483e8 (1000)
lines 83-105/105 (END)

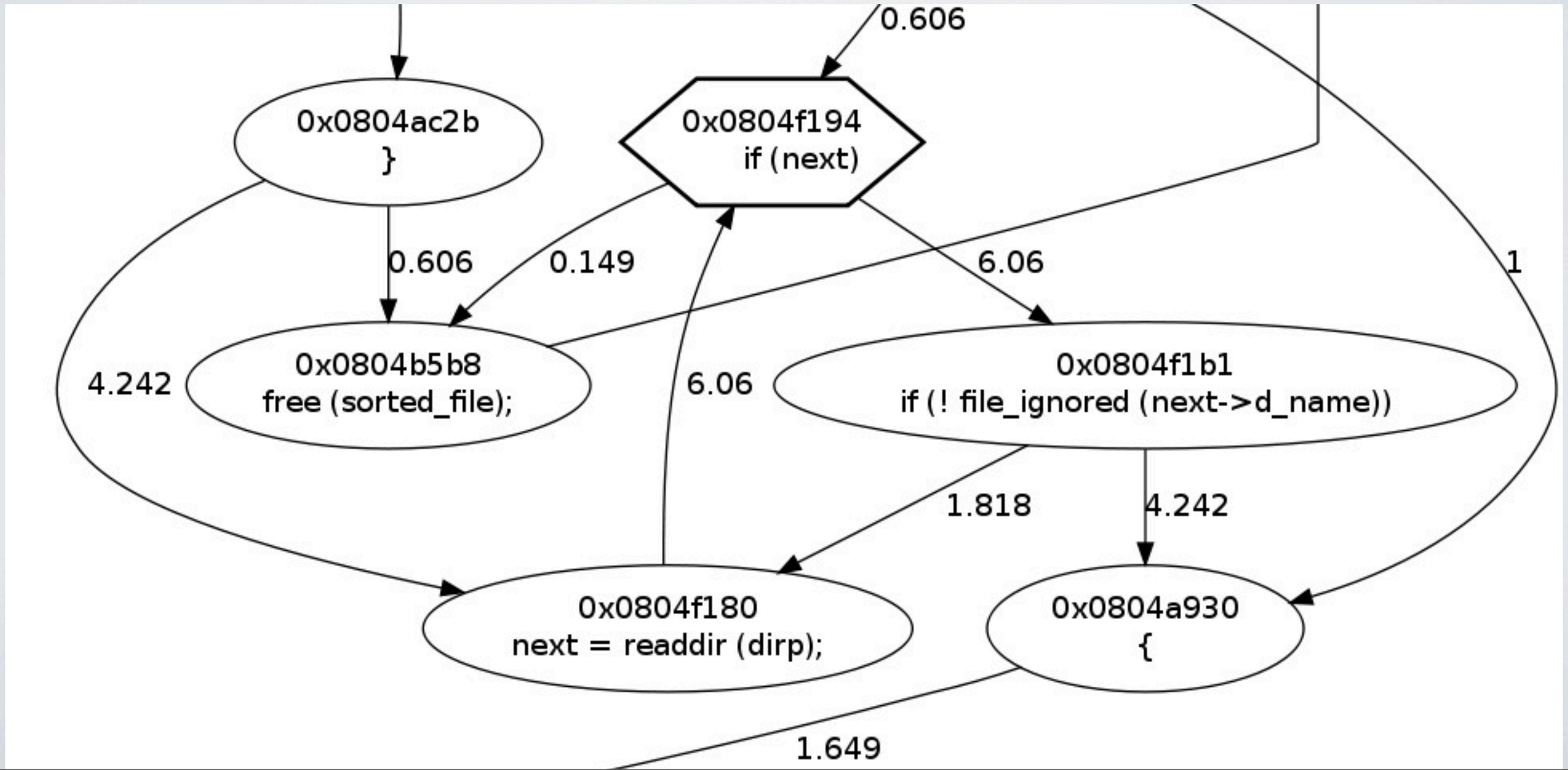

xterm
tnathan@selkirk:loopgrind$ ./loopgrind.sh tests/simpleloop | perl tools/analyze.
pl tests/simpleloop


xterm
Building graph for displaying.....done.
Max in-weight: 19.000 (0x080483c7)
Address of loop: 0x080483c7
Dist from main: 1
lines 1-4/4 (END)

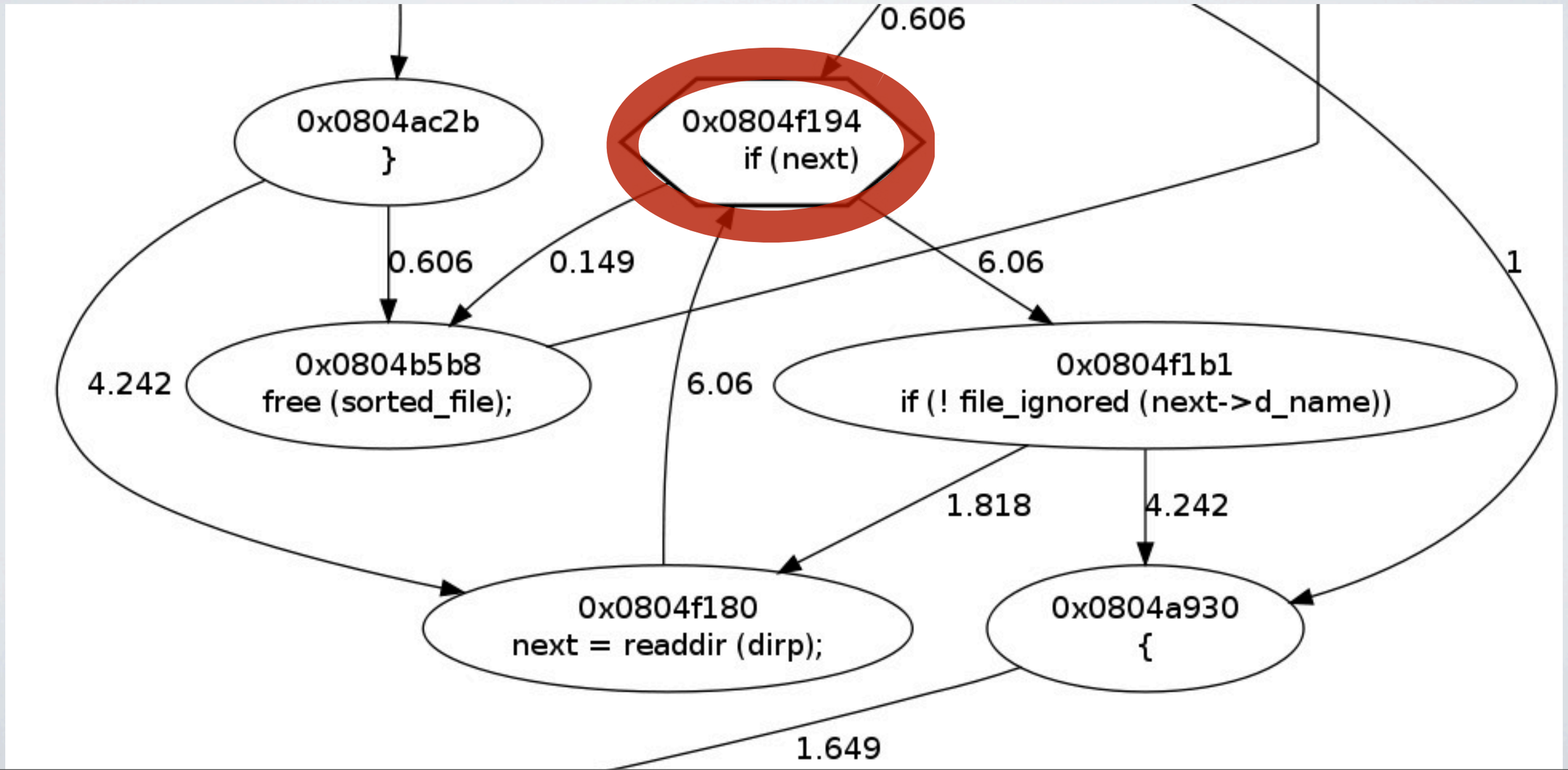
SECOND GOAL

"What changes within iterations of a main loop?"

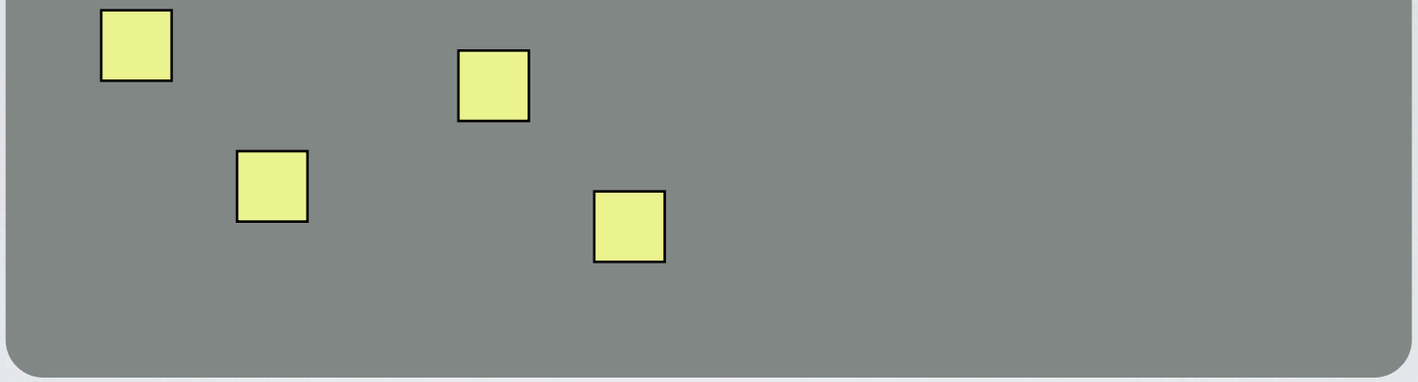
Process' address space

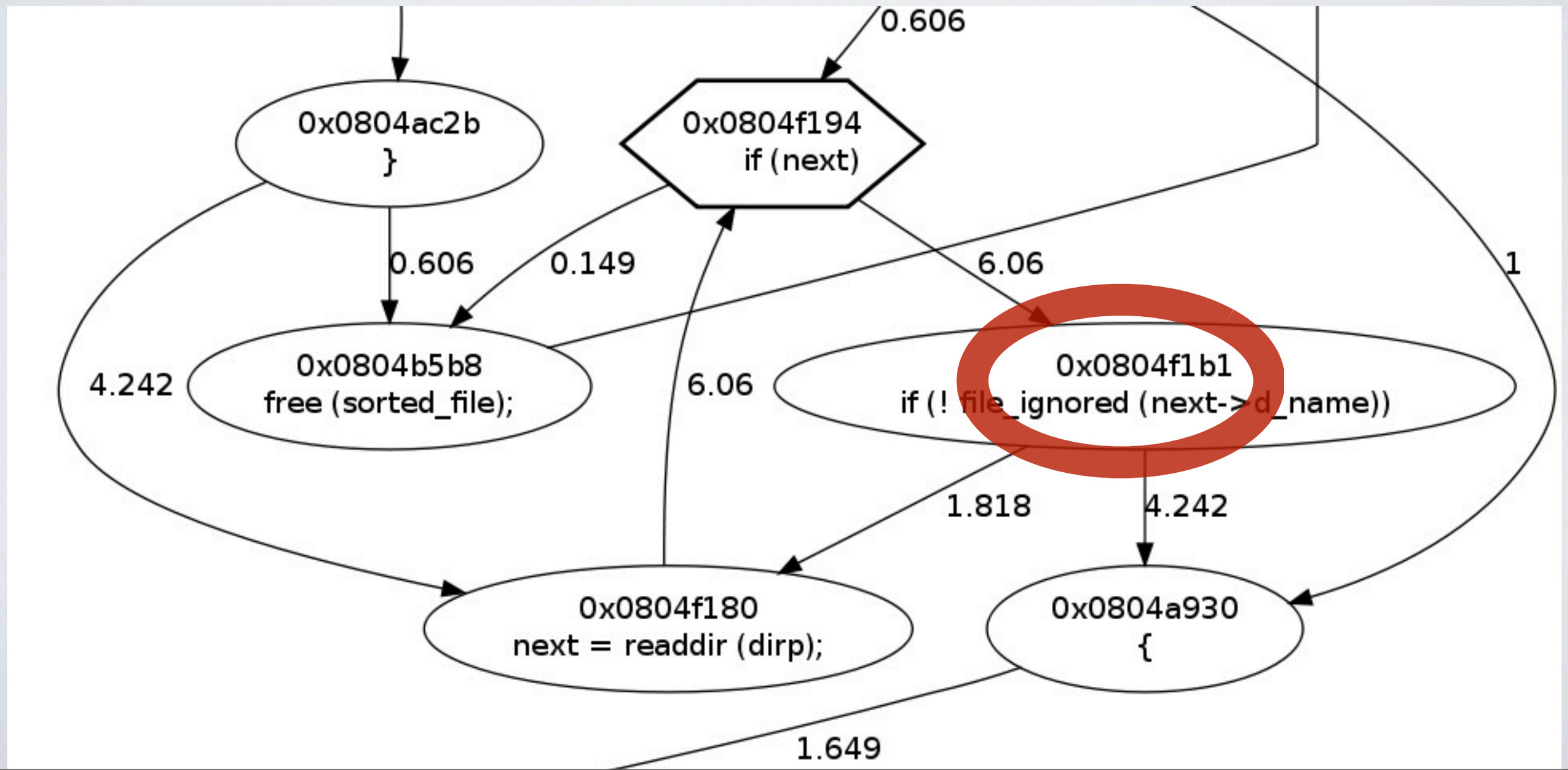


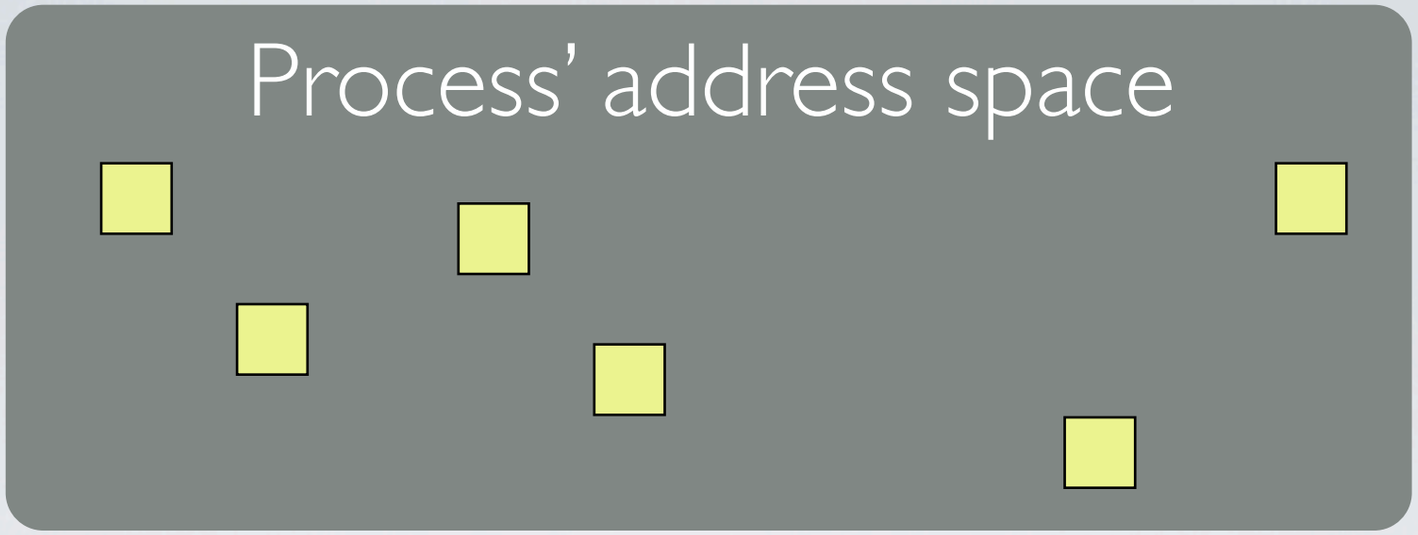
Process' address space

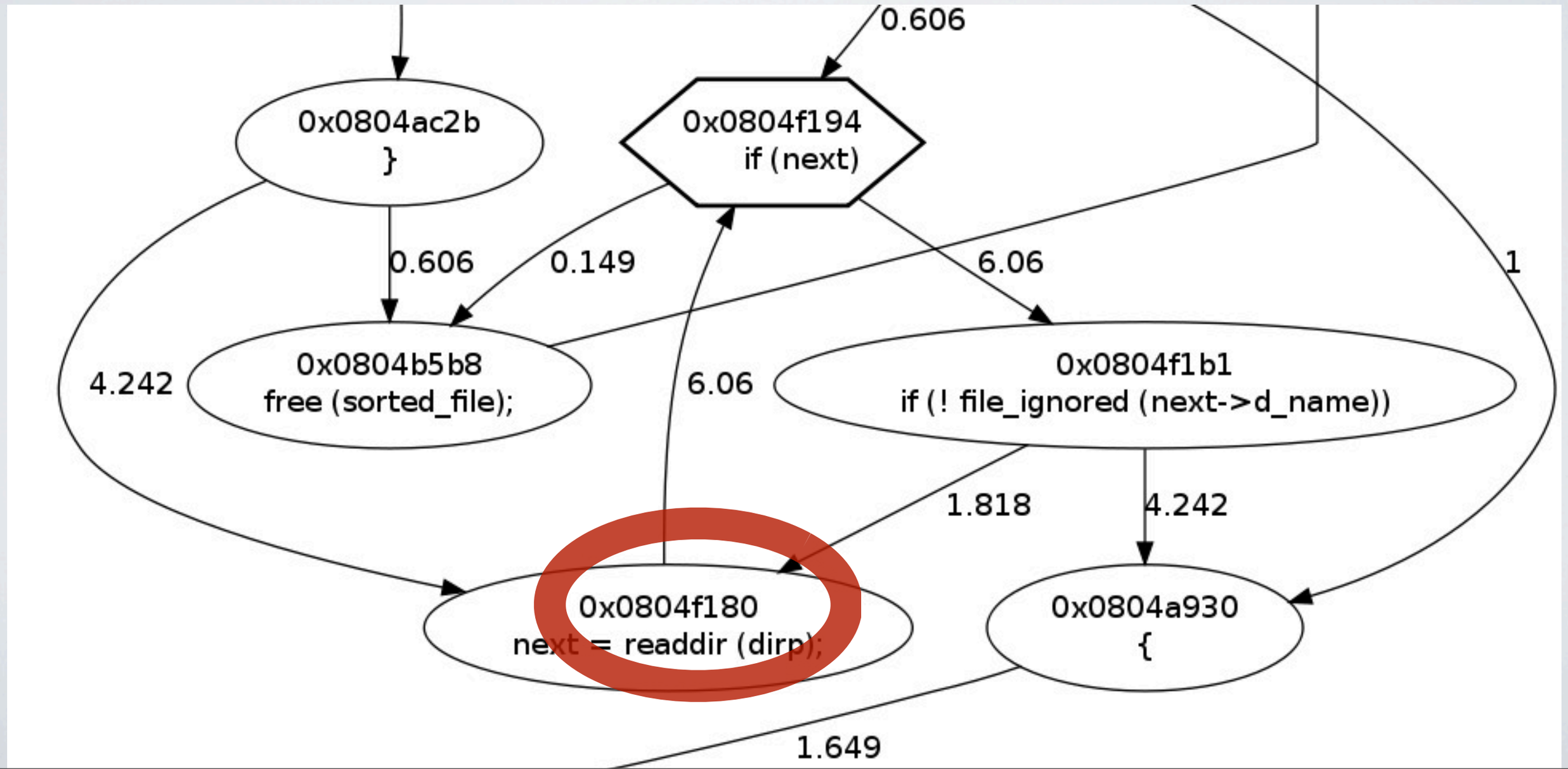


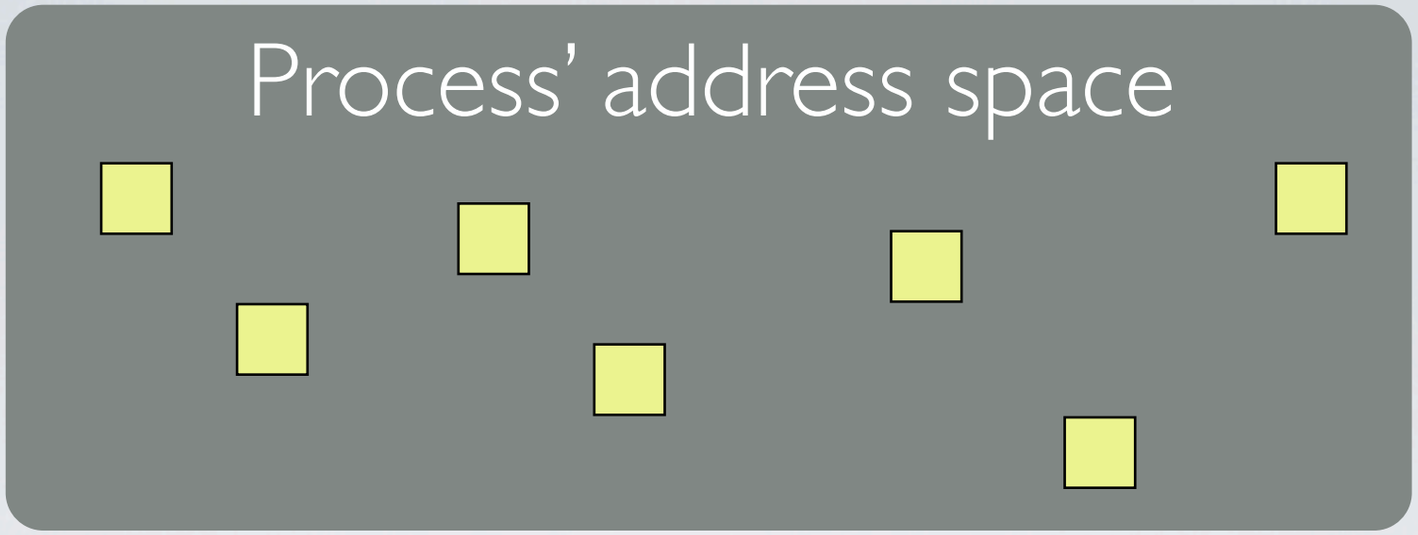
Process' address space

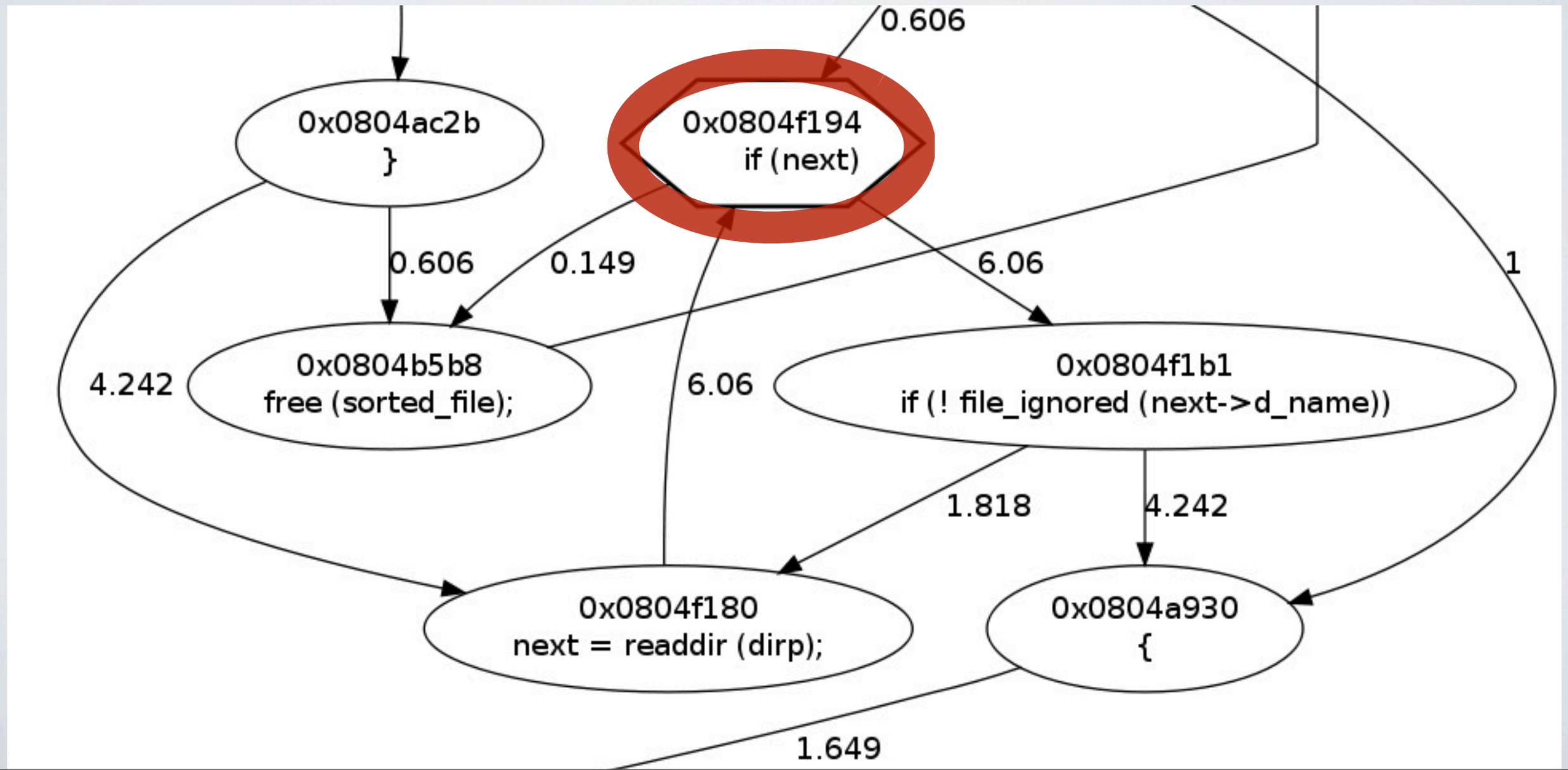




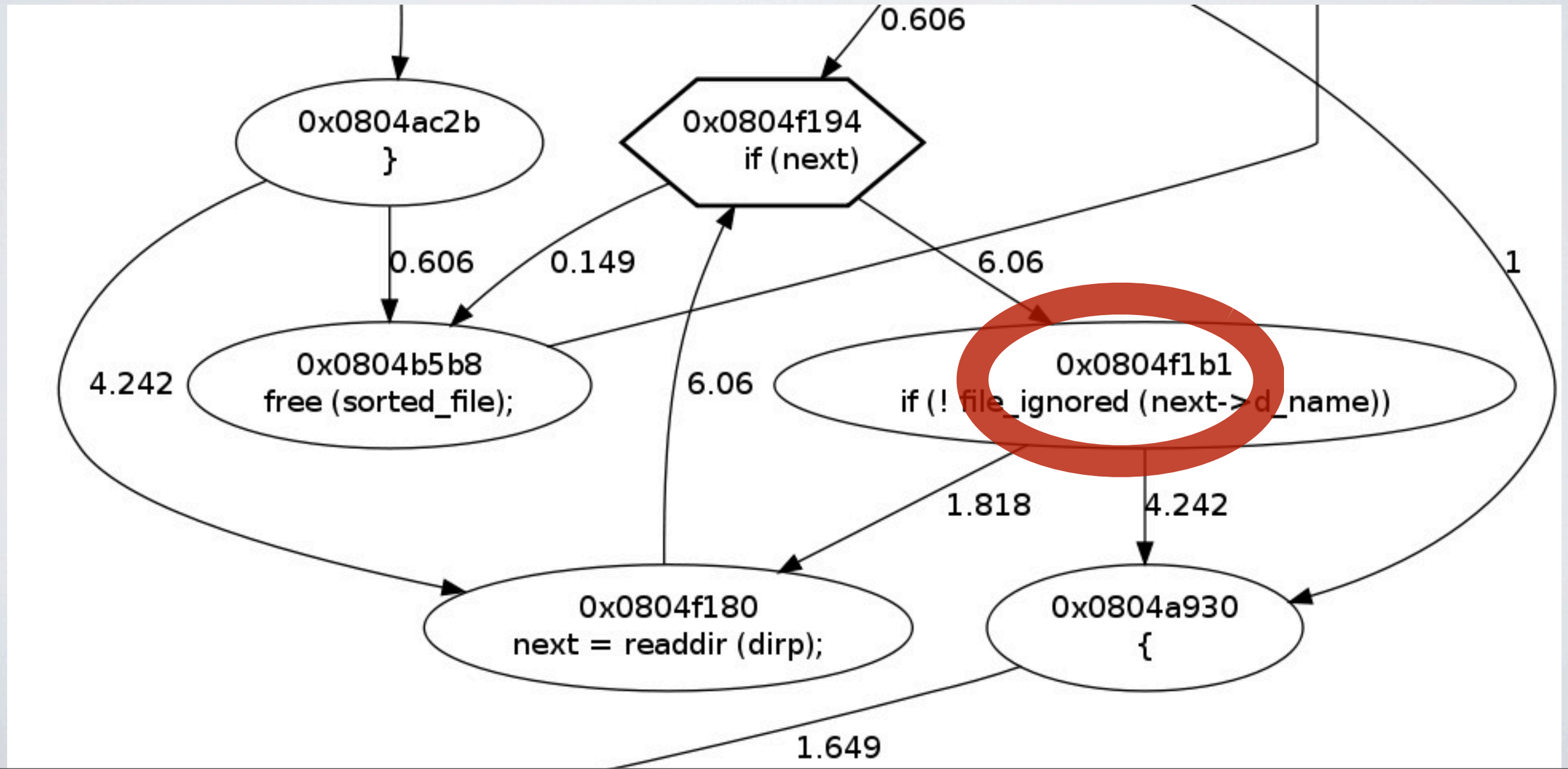

Process' address space


0.606
0x0804ac2b
}
0x0804f194
if (next)
0.606
0.149
6.06
1
4.242
0x0804b5b8
free (sorted_file);
6.06
0x0804f1b1
if (! file_ignored (next->d_name))
1.818
4.242
0x0804f180
next = readdir (dirp);
0x0804a930
{
1.649

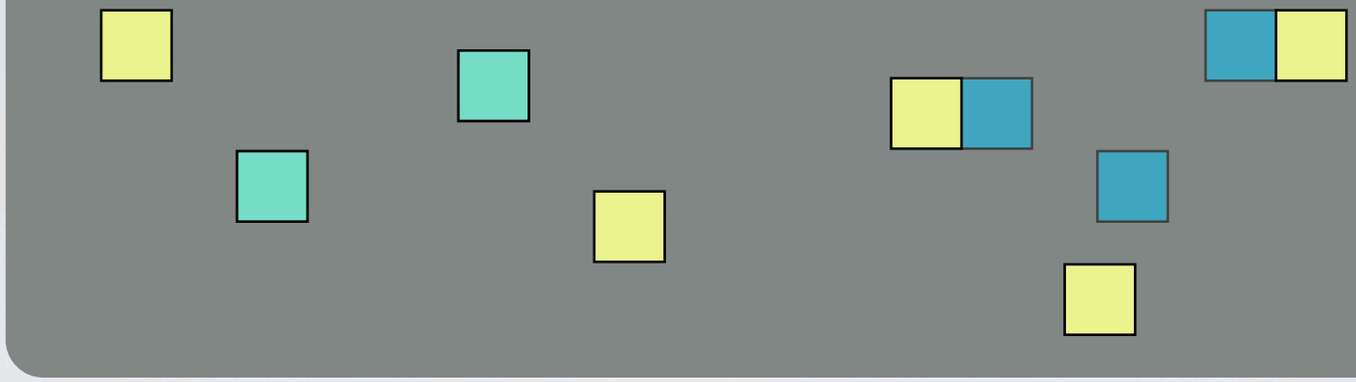

Process' address space

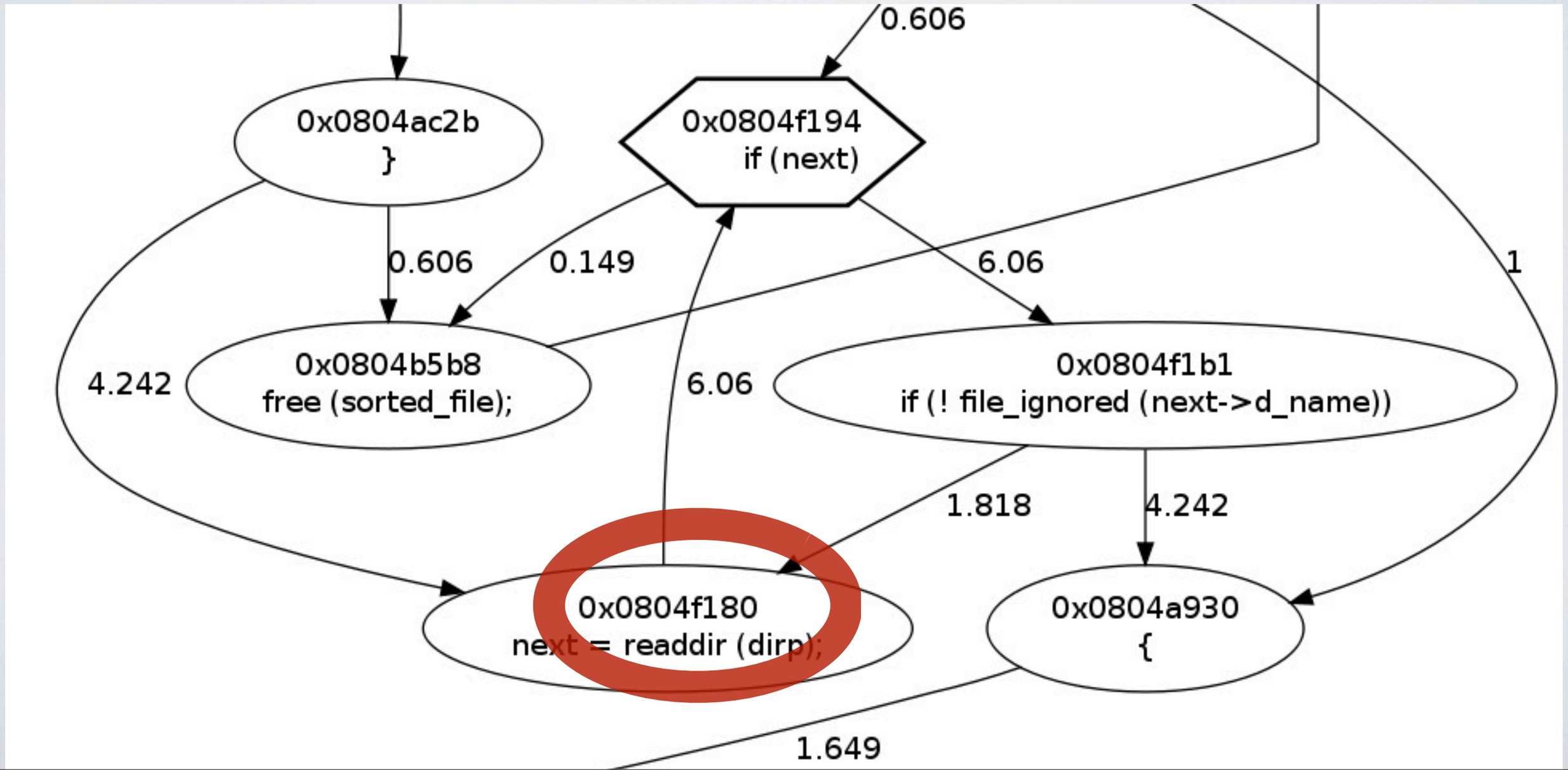

0.606
0x0804ac2b
}
0x0804f194
if (next)
0.606
0.149
6.06
1
4.242
0x0804b5b8
free (sorted_file);
6.06
0x0804f1b1
if (! file_ignored (next->d_name))
1.818
4.242
0x0804f180
next = readdir (dirp);
0x0804a930
{
1.649


Process' address space

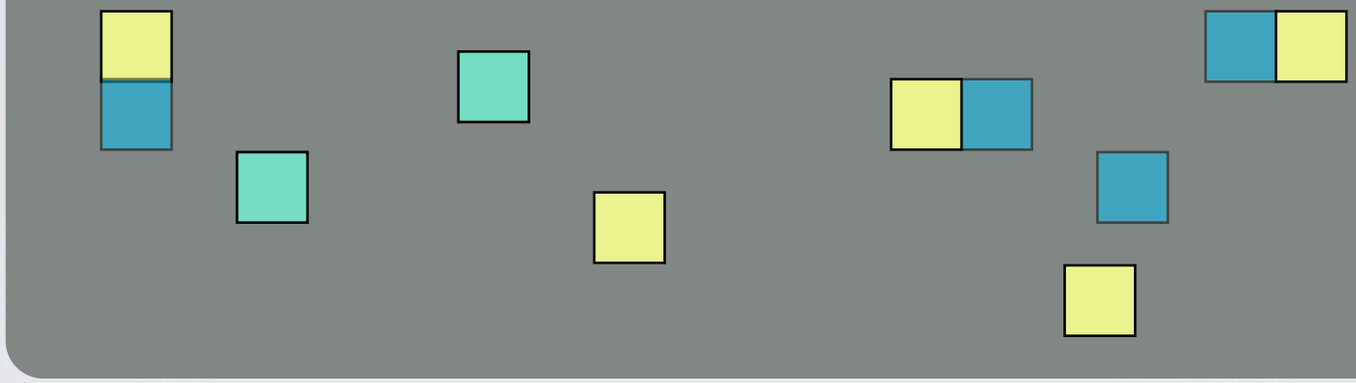

0.606
0x0804ac2b
}
0x0804f194
if (next)
0.606
0.149
6.06
1
4.242
0x0804b5b8
free (sorted_file);
6.06
0x0804f1b1
if (! file_ignored (next->d_name))
1.818
4.242
0x0804f180
next = readdir (dirp);
0x0804a930
{
1.649

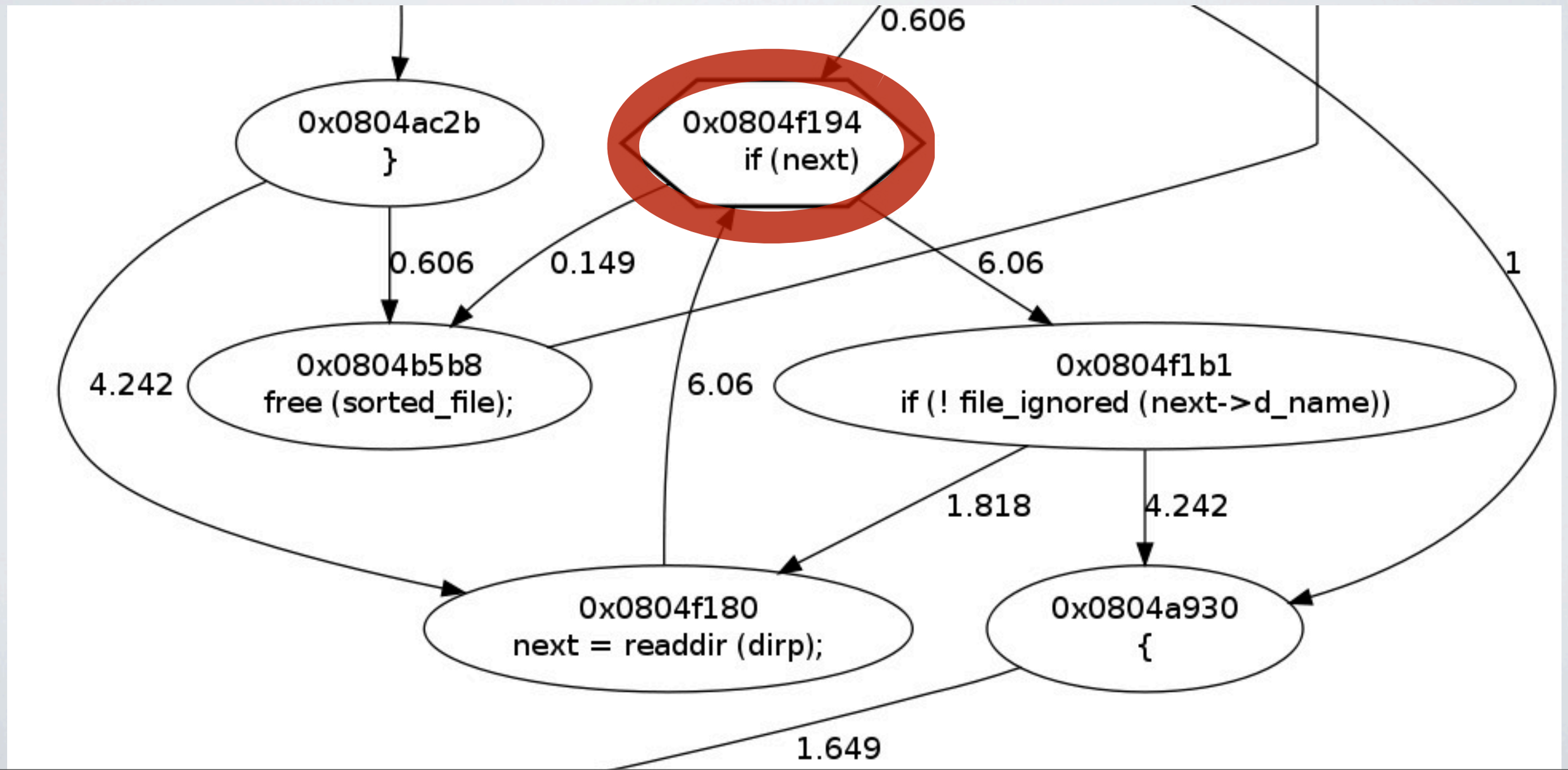
Process' address space





Process' address space

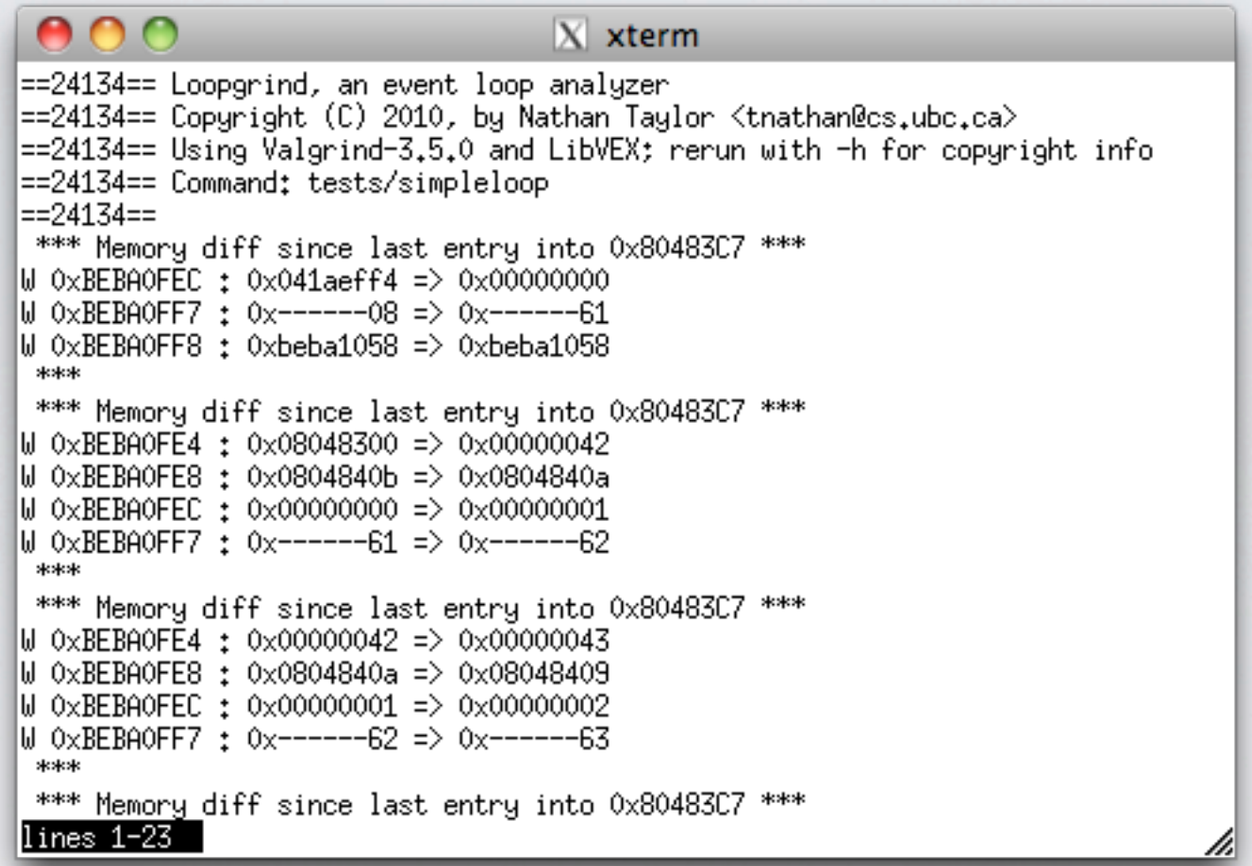




MEMORY DIFFS OVER LOOPS

- Tool summarizes changes in memory upon reentering loop
- Needs work before it's ready for prime time - tends to die within the VEX IR runtime w.r.t. floating point (ie. `atof()`)


xterm
tnathan@selkirk:loopgrind$ ./loopgrind.sh -a 0x080483c7 tests/simpleloop | less


xterm
==24134== Loopgrind, an event loop analyzer
==24134== Copyright (C) 2010, by Nathan Taylor <tnathan@cs.ubc.ca>
==24134== Using Valgrind-3.5.0 and LibVEX; rerun with -h for copyright info
==24134== Command: tests/simpleloop
==24134==
*** Memory diff since last entry into 0x80483C7 ***
W 0xBEBA0FEC : 0x041aeff4 => 0x00000000
W 0xBEBA0FF7 : 0x------08 => 0x------61
W 0xBEBA0FF8 : 0xbeba1058 => 0xbeba1058
***
*** Memory diff since last entry into 0x80483C7 ***
W 0xBEBA0FE4 : 0x08048300 => 0x00000042
W 0xBEBA0FE8 : 0x0804840b => 0x0804840a
W 0xBEBA0FEC : 0x00000000 => 0x00000001
W 0xBEBA0FF7 : 0x------61 => 0x------62
***
*** Memory diff since last entry into 0x80483C7 ***
W 0xBEBA0FE4 : 0x00000042 => 0x00000043
W 0xBEBA0FE8 : 0x0804840a => 0x08048409
W 0xBEBA0FEC : 0x00000001 => 0x00000002
W 0xBEBA0FF7 : 0x------62 => 0x------63
***
*** Memory diff since last entry into 0x80483C7 ***
lines 1-23

CASE STUDY

CASE STUDY

- Pascal compiler and JVM-like interpreter

CASE STUDY

- Pascal compiler and JVM-like interpreter
  - First part: compiling a 140 line Pascal program

CASE STUDY

- Pascal compiler and JVM-like interpreter
  - First part: compiling a 140 line Pascal program
  - Second part: Executing it within the runtime environment, outputting the above

COMPILER TEST

TWO LOOPS OF INTEREST

```
/home/ntaylor/code/loopgrind/tests/ceiling/src/lex.yy.c:946
 804dad1:       0f b6 03                movzbl (%ebx),%eax
 804dad4:       0f b6 c0                movzbl %al,%eax
 804dad7:       8b 04 85 60 ab 05 08    mov    0x805ab60(,%eax,4),%eax
 804dade:       88 45 c3                mov    %al,-0x3d(%ebp)
```

```
943 yy_match:
944         do
945             {
946             register YY_CHAR yy_c = yy_ec[YY_SC_TO_UI(*yy_cp)];
947             if ( yy_accept[yy_current_state] )
948                 {
949                 (yy_last_accepting_state) = yy_current_state;
950                 (yy_last_accepting_cpos) = yy_cp;
951                 }
952             while ( yy_chk[yy_base[yy_current_state] + yy_c] != yy_current_state )
953                 {
954                 yy_current_state = (int) yy_def[yy_current_state];
955                 if ( yy_current_state >= 139 )
956                     yy_c = yy_meta[(unsigned int) yy_c];
957                 }
958             yy_current_state = yy_nxt[yy_base[yy_current_state] + (unsigned int) yy_c];
959             ++yy_cp;
960             }
961         while ( yy_base[yy_current_state] != 230 );
```

COMPILER TEST

TWO LOOPS OF INTEREST

```
/home/ntaylor/code/loopgrind/tests/ceiling/src/codegen.c:172
 805370d:       a1 4c 01 06 08          mov    0x806014c,%eax
 8053712:       85 c0                   test   %eax,%eax
 8053714:       75 26                   jne    805373c <emitCode+0x4b>
 8053716:       a1 50 01 06 08          mov    0x8060150,%eax
 805371b:       85 c0                   test   %eax,%eax
 805371d:       75 1d                   jne    805373c <emitCode+0x4b>
```

```
162 /*
163  * emitCode takes a string and prints it to the .asc file assuming it is code
164  * will emit \n.
165  */
166 void emitCode(char *code)
167 {
168         /* write to file if we haven't failed in pal so far or in const*/
169         if(queue) {
170                 addQueueCode(code);
171         }
172         else if (!failed && !noemit_flag) {
173                 fprintf(asc_file, "\t%s\n", code);
174
175         }
176 }
```

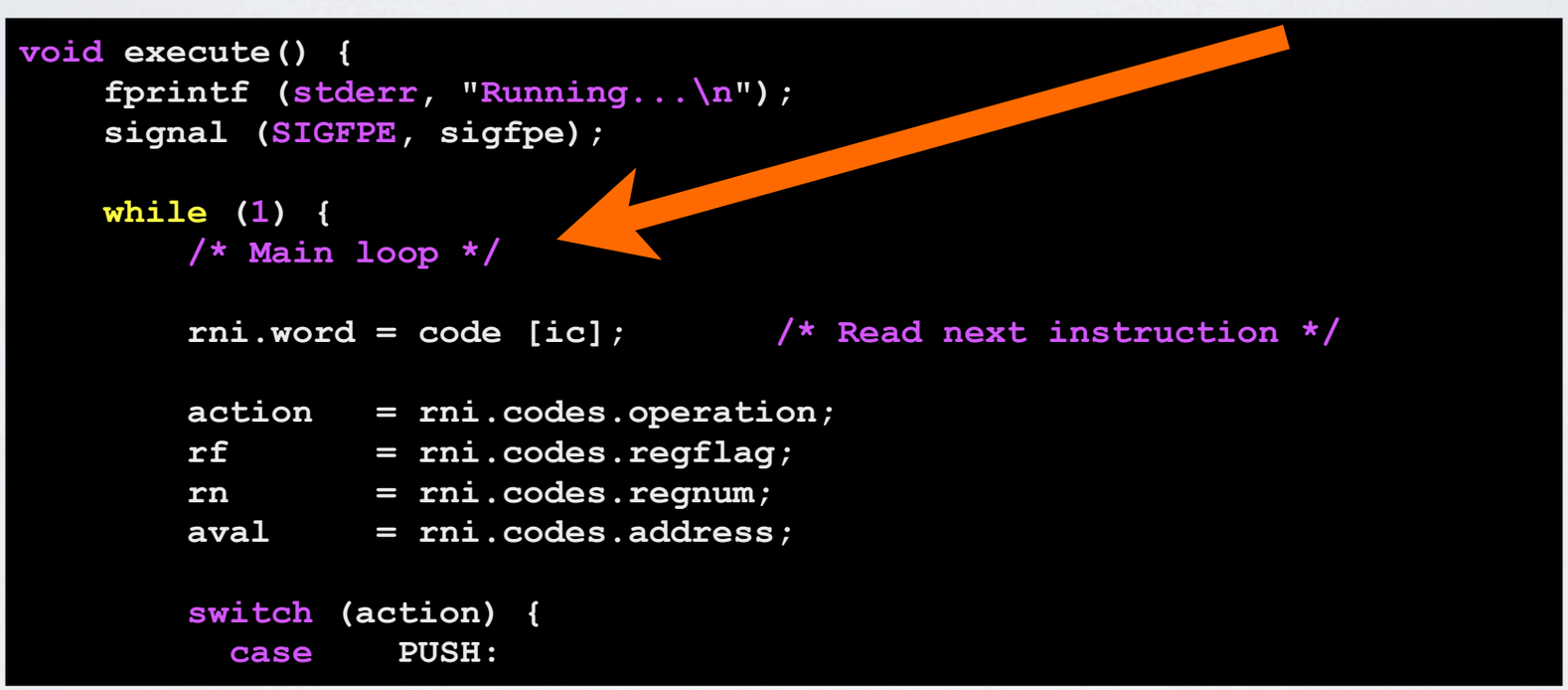
INTERPRETER TEST

```
804b6a0:       ff 05 68 e4 04 08       incl   0x804e468
804b6a6:       e9 3e e9 ff ff          jmp    8049fe9 <execute+0x83>
```

```c
void execute() {
    fprintf (stderr, "Running...\n");
    signal (SIGFPE, sigfpe);

    while (1) {
        /* Main loop */

        rni.word = code [ic];       /* Read next instruction */

        action   = rni.codes.operation;
        rf       = rni.codes.regflag;
        rn       = rni.codes.regnum;
        aval     = rni.codes.address;

        switch (action) {
          case    PUSH:
```

INTERPRETER TEST

```
804b6a0:        ff 05 68 e4 04 08       incl   0x804e468
804b6a6:        e9 3e e9 ff ff          jmp    8049fe9 <execute+0x83>
```



```
void execute() {
    fprintf (stderr, "Running...\n");
    signal (SIGFPE, sigfpe);

    while (1) {
        /* Main loop */

        rni.word = code [ic];        /* Read next instruction */

        action   = rni.codes.operation;
        rf       = rni.codes.regflag;
        rn       = rni.codes.regnum;
        aval     = rni.codes.address;

        switch (action) {
          case    PUSH:
```

INTERPRETER TEST

```
804b6a0:        ff 05 68 e4 04 08       incl   0x804e468
804b6a6:        e9 3e e9 ff ff          jmp    8049fe9 <execute+0x83>
```

```
void execute() {
    fprintf (stderr, "Running...\n");
    signal (SIGFPE, sigfpe);

    while (1) {
        /* Main loop */

        rni.word = code [ic];        /* Read next instructio

        action   = rni.codes.operation;
        rf       = rni.codes.regflag;
        rn       = rni.codes.regnum;
        aval     = rni.codes.address;

        switch (action) {
          case    PUSH:
```

FUTURE WORK

- Make more robust to scary real-world input
- Better metric for loop start
- Analysis of output! Tons of ideas here....
    - Latent semantic indexing for program classification?
    - Loop diffs as states in a hidden Markov model?
- Integration into a full system simulator such as QEMU?


0.596
0x08057f20
{
2.081
0.472
0x08057f32
if (!p && n != 0)
3.39
0x08057f96
}
0.606
0.755
1.7
0x0804b4d5
for (i = 0; i < cwd_n_used; i++)
1.104
0x08057f75
}
0.755
1.212
0.622
0x0804b4f5
if (sort_type == sort_none)
0x0805513f
errno = e;
0x08057fa1
}
0x0804f7e0
{
0.606
0.703
0.755
0.606
0.606
0x08054d20
{
0.315
0x080494b0
ff 25 04 10 06 08 jmp *0x8061004
1.1
0.606
1.649
0.149
0x08054d36
if (n == 2)
0.15
0x08055127
struct quoting_options *p = xmemdup (o ? o : &default_quoting_options,
0x0804a3c0
cwd_n_used = 0;
1.1
0.356
0x08054d3c
}
0.315
0.606
0.149
0.606
6.06
0x0804b5b8
free (sorted_file);
0x0804f1b1
6.06
1.818
0.315
0x0804f180
next = readdir (dirp);
2.346
0.617
1.247
0.466
0.151
0.315
0.164
0.606
0.315
0.932
0.302
0.164
0.315
0x08054e6d
0x08054df5
0x08054e48
base[i++] = bb;
0x08054dd0
b++;
1.649
0x0804a9bd
f->stat.st_ino = inode;
1.043
0x0804ac45
if (command_line_arg
0.606
1.043
0x08049810
ff 25 dc 10 06 08 jmp *0x80610dc
0.151
0.164
0.315
0x08054f11
if (b == blim)
0x08054ebd
}
0x0804abfa
f->name = xstrdup (name);
0.149
0.151
0x0804b5ad
}
0x08054d6c
if (n1 < 2)
0.155
0x0804c9f8
if (print_block_size)
0
0x080551d0
{
0
0x080551e7

THANKS