CS 35L Software Construction Laboratory (Lab8-A)
Mon, Feb 27, 2012, Ver 1.1

Process Revisited

```
  Process: Program + Data:
  Memory of a precess can be devided into three regions:
  1) Text: programs and read-only data
           (loaded from text section of a executable file)
  2) Data: stores static variables
           (loaded from bss section of a executable file)
  3) Stack: stores dynamic varialbles, function frames
```

Function Call

```
  +----------------------------+
   void main() {
     function(1, 2, 3);
   }
  +----------------------------+
  pushl $3         # push arguments from right to left
  pushl $2         # when they are poped out the order is left to right
  pushl $1
  call function    # jump to the address of the callee function
                   # call will push the current IP register (instruction pointer)
                   # into the stack.
                   # This is the RET address for the callee function
  pushl %ebp       # EBP: the base pointer of the current function's stack
                   #      i.e. the start boundary of the current function
  movl %esp,%ebp   # ESP: the stack pointer of the current function's stack
  subl $20,%esp    #      i.e. the cutting edge of the current function
```

Buffer: a contiguous block of computer memory that holds multiple instances
of the same data type.

- -- static buffer: allocated at load time on the data segment.
- -- dynamic: allocated at run time on the stack.

Buffer Overflow

the result of stuffing more data into a buffer than it can handle.
sample code:

```
  +----------------------------+
   void function(char *str) {
     char buffer[16];

     strcpy(buffer,str);
   }

   void main() {
     char large_string[256];
     int i;

     for( i = 0; i < 255; i++)
       large_string[i] = 'A';

     function(large_string);
```

```
 }
+----------------------------+
bottom of                                                    top of
memory                                                       memory
               buffer           sfp   ret   *str
<------       [               ][    ][    ][    ]

top of                                                     bottom of
stack                                                          stack
```

IMPORTANT: buffer overflow allows us to change the return address of a function
More Reading:
http://insecure.org/stf/smashstack.html

Getting start with Lab 8

1. Grab the tarball, untar it
2. Apply the patch given from the course website
3. Run "./configure", "make" and "make install"
   -- "make install" is optional, if you wanna install it, you may face some
      problem related to unexisted user and group.
4. Fix other bugs if necessary
5. If you simply run "./thttpd" it will not work.
6. Read MANPAGE of thttpd carefully. Especially, pay attention to these options:
   -C, specifies a config-file for thttpd
   -p, set port
   -d, set root directory
7. If you are still confused, you may find the following link helpful:
   http://www.acme.com/software/thttpd/notes.html
     You do not need to go through everything in that note page, but i helps to
   have a feeling about how an http server should be configured.