Guide to Using Signals

- UNIX system uses signals to notify a process that a particular event has occurred. Signals may be either synchronous or asynchronous, depending upon the source and the reason for the event being signalled. Once a signal has been generated by the occurrence of a certain event (e.g., division by zero, illegal memory access, user entering <Control><C>, etc.), the signal is delivered to a process where it must be handled. A process receiving a signal may handle it by one of the following techniques:
  - ignoring the signal
  - using the default signal handler, or
  - providing a separate signal-handling function.
-
- Signals may be handled by first setting certain fields in the C structure `struct sigaction` and then passing this structure to the `sigaction()` function. Signals are defined in the include file `/usr/include/sys/signal.h`. For example, the signal `SIGINT` represents the signal for terminating a program with the control sequence <Control><C>. The default signal handler for `SIGINT` is to terminate the program. Alternatively, a program may choose to set up its own signal-handling function by setting the `sa_handler` field in `struct sigaction` to the name of the function which will handle the signal and then invoking the `sigaction()` function, passing it (1) the signal we are setting up a handler for, and (2) a pointer to `struct sigaction`.
-
- Below, is a C program that uses the function `handle_SIGINT()` for handling the `SIGINT` signal. This function prints a message and then invokes the `exit()` function to terminate the program. (We must use the `write()` function for performing output rather than the more common `printf()` as the former is known as being signal-safe, indicating it can be called from inside a signal-handling function; such guarantees cannot be made of `printf()`.) This program will run in the `while (1)` loop until the user enters the sequence <Control><C>. When this occurs, the signal-handling function `handle_SIGINT()` is invoked.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>

#define BUFFER_SIZE 50
static char buffer[BUFFER_SIZE];

/* Signal handler function */
void handle_SIGINT()
{
      write(STDOUT_FILENO, buffer, strlen(buffer));
      exit(0);
}
```

```
int main(int argc, char *argv[])
{
        /* set up the signal handler */
        struct sigaction handler;
        handler.sa_handler = handle_SIGINT;
        handler.sa_flags = 0;
        sigemptyset(&handler.sa_mask);
        sigaction(SIGINT, &handler, NULL);

        strcpy(buffer,"I caught a Ctrl-C!\n");

        printf("Program now waiting for Ctrl-C.\n");
        while (1)
              ;

        return 0;
}
```

- The signal-handling function should be declared above `main()` (unless using prototypes) and because control can be transferred to this function at any point, no parameters may be passed to it this function. Therefore, any data that it must access in your program must be declared globally, i.e. at the top of the source file before your function declarations.