```c
//----------------------------------------------------------------------
// SOPE - 2013/2014 - JAS
// Signals - s10.c
// Illustrating some synchronous signals

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

void sigsegv_handler(int signo)
{
      printf("In SIGSEGV handler\n");
      printf("Error: forbidden memory access !!!\n");
      exit(1); // "return" => infinite cycle
}

int main(void)
{
       char *name = "Ana Sousa";
      //UNCOMMENT THE 2 ALTERNATIVES
      /*
      if (signal(SIGSEGV,SIG_IGN)==SIG_ERR)
      {
        printf("SIGSEGV can't be ignored ...!\n");
        exit(1); // no error, but it can't be ignored
      }
      */
      //signal(SIGSEGV, sigsegv_handler);

       //UNCOMMENT ONE OF THE 2 ALTERNATIVES IN TURN
      // name[1]='d'; // write to read-only memory address
      int  *year; year = (int *) 0; *year = 2014; // invalid memory address

      return 0;
}
```

```c
//------------------------------------------------------------------------
// SOPE - 2012/2013 - JAS
// Signals - s11.c
// Basic process synchronization
// Father and son write "Hello world!":
// father writes "Hello", son writes "world!"

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

//int received_signal = 0;

void sigusr1_handler(int signo)
{
	//	received_signal = 1;
	return;
}

int main(void)
{
	pid_t pid;
	int status;

	pid=fork();
	if (pid > 0)
	{
		printf("Hello ");
		fflush(stdout); // <----- NOTE THIS
		sleep(1);       // <----- NOTE THIS / NOT VERY GOOD SOLUTION ...
		// without sleep(), child could die before installing handler
		// ALTERNATIVE: install handler before forking
		kill(pid,SIGUSR1);
		wait(&status);
		exit(0);
	}
	else
	{
		signal(SIGUSR1,sigusr1_handler);
		pause();
		// TEST THE FOLLOWING ALTERNATIVE
		// after uncommenting above received_signal = ... statements
		//while (!received_signal) sleep(1);
		printf("world ! \n");
		exit(0);
	}
}
```