```
/**
 * File: pbreak.in
 *       Extension 2019-06-12_22 test program.
 *
 * Author: Rambod Rahmani <rambodrahmani@autistici.org>
 *         Created on 22/09/2019.
 */

#include <sys.h>
#include <lib.h>

/**
 *
 */
process bad1 body bad(1), 20, LIV_UTENTE;

/**
 *
 */
process bad2 body bad(2), 8, LIV_UTENTE;

/**
 *
 */
process bad4 body badb(3), 21, LIV_UTENTE;

/**
 *
 */
process bad5 body badb(4), 22, LIV_UTENTE;

/**
 *
 */
process usr1 body usr(1), 5, LIV_UTENTE;

/**
 *
 */
process usr2 body usr(2), 15, LIV_UTENTE;

/**
 *
 */
process usr3 body usr(3), 3, LIV_UTENTE;

/**
 *
 */
process dbg1 body debugger(0), 10, LIV_UTENTE;

/**
 *
 */
process dbg2 body debugger(1), 11, LIV_UTENTE;

/**
 *
 */
process dbg3 body debugger(2), 4, LIV_UTENTE;

/**
 *
 */
process last body last_body(0), 1, LIV_UTENTE;

semaphore sync value 0;

/**
 * Bad process body 1: calls the int3 instruction without using the breakpoint()
```

```
 * primitive.
 */
process_body bad(int a)
{
    asm("int3");
    printf("processo errato %d", a);
}

/**
 *
 */
void catch_me(int a)
{
    printf("proc%d: eseguo funzione", a);
}

/**
 *
 */
vaddr bad_addr[] = { 1000, 0xffffc00000000000 };

/**
 * Bad process body 2: calls the breakpoint() primitive with the wrong address.
 * The breakpoint() primitive can be called only from addresses belonging to the
 * user process shared memory area.
 */
process_body badb(int a)
{
    breakpoint(bad_addr[a - 3]);
    printf("processo errato %d", a);
}

/**
 * User process:
 */
process_body usr(int a)
{
    if (a % 2 == 0)
    {
        sem_wait(sync);
    }

    printf("proc%d: prima della funzione", a);

    catch_me(a);

    printf("proc%d: dopo la funzione", a);
}

/**
 *
 */
process_body debugger(int a)
{
    printf("debugger %d: chiamo breakpoint", a);

    natl proc = breakpoint(reinterpret_cast<natq>(catch_me));

    if (proc == 0xFFFFFFFF)
    {
        printf("debugger %d: occupato", a);
        }
    else
    {
        sem_signal(sync);
        printf("debugger %d: breapoint intercettato, processo: %d", a, proc);
    }
}

/**
```

```
 *
 */
process_body last_body(int a)
{
    pause();
}
```