Sebastian Mountaniol for Tonian

This report is result of programming exercise from Tonian.com

## 1. Realization of:

```
extern sg_entry_t *sg_map(void *buf, int length);
extern void sg_destroy(sg_entry_t *sg_list);
extern int sg_copy(sg_entry_t *src, sg_entry_t *dest, int src_offset, int count);
```

See attached file sg_copy.c

2. Find all possible mistakes (there might be none) in the following piece of code. The function apparently takes an RTP packet, and creates a new RTP packet with all the fields same as the previous RTP packet except that the sequence number and timestamp are changed.

All my remarks colored by red color.

```
struct rtp_packet {
        int version:2;
        int padding:1;
        int extension:1;
        int ccount:4;
        int marker:1;
        int payloadtype:7;
        int sequence:16;
        int ssrc;
        int timestamp;
        int payload_length;

        char *payload;
};

/* Fixing errors of in-function definitions */
typedef struct rtp_packet rtp_packet;

void *create_packet(void *p)
{

rtp_packet *ptr; /* Error: must be struct rtp_packer * ptr - fixed by adding typedef  */
rtp_packet *dummy; /* Error: must be struct rtp_packet * dummy */

ptr = (rtp_packet *) p;

dummy = (rtp_packet *)malloc(1, sizeof(rtp_packet *));
/* here are 2 errors, fixed in the next lines:
* Error 1: the malloc syntax requires only one argument, the size of rquired memory
* Error 2: here should be allocation of the structure, not a pointer to the structure */
* dummy = (rtp_packet *) malloc(sizeof(rtp_packet)); */

if (dummy == NULL) {
        return NULL;
}

/* Error 1. Logical error.
 * Error 2. Memory leak */

*dummy = *ptr;

/* Error: freeing the external buffer (void *p)
 * Here are several problems:
 * 1. This is destroy the original buffer, what is wronh, but...
 * 2. This buffer can be not allocated by malloc family, but be just regular variable passed by
address, like:
 *
 * struct rtp_packet packet;
 * create_packet(&packet); <-- crash*/

free(ptr);

/* ptr now points to a junk */
ptr = dummy;

/* crash of the application or kernel: trying to free a random memory */
free(dummy);

 /* Meaningless, the application / kernel have been crashed. */
ptr->sequence ++;
ptr->timestamp += 160;

/* Nevermore here */
return ptr;
}
```

In the attached snipped2.c I fixed all bugs. See there correct function.

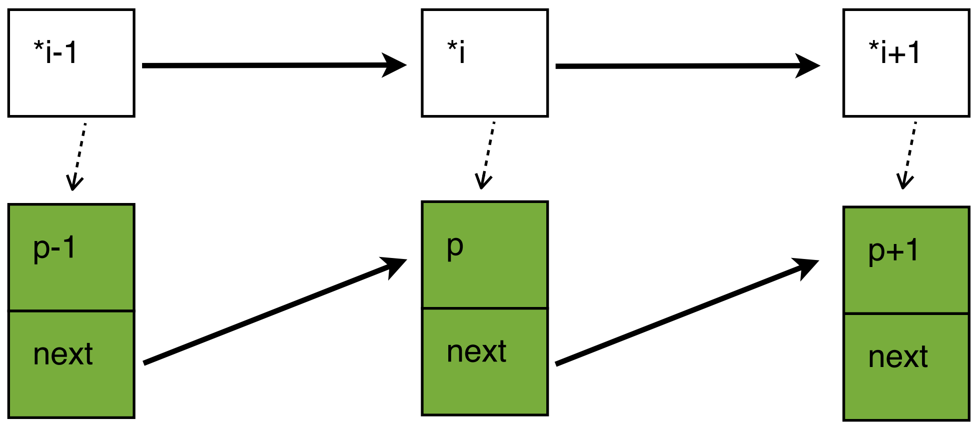
3. Pointer manipulation: The following function is suppose to delete an element from a singly linked list. Will it work? If yes, show the steps when the linked list has initially 4 elements and the last element is supposed to be deleted. If not, point out a mistake.

My remarks are red.

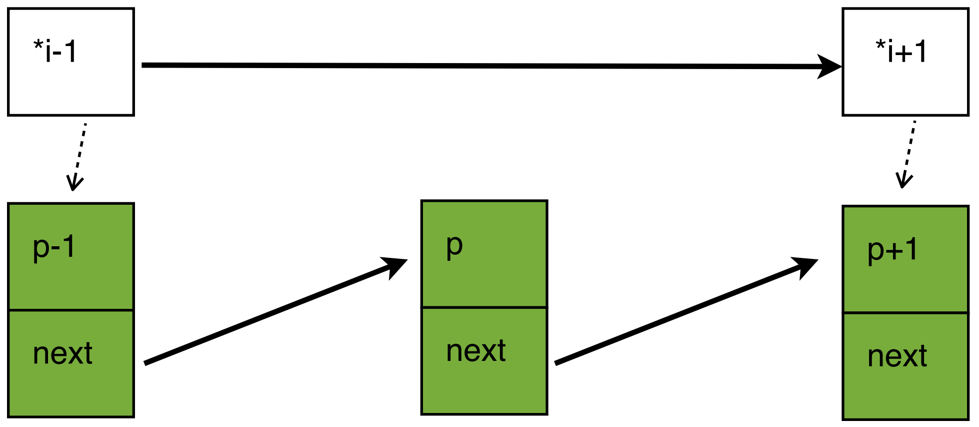
```
/*
 * remove_from_pending_list: removes the pkt from the list when
 * it is no longer needed. pending_list is a global variable.
 */
void remove_from_pending_list(packet *p)
{
        /* Here is the problem number 1: the '**i' points to the memory where 'pending_list' pointer is kept
         * So if changing the value of *i is the same as changing value of 'pending_list' */

        packet **i = &pending_list;

        for (;(*i) != NULL && ((*i) != p); *i = ((*i)->next)) {
                /* do nothing */;
        }

        /* Now '*i' points to the 'p'. As well, the 'pending_list' now doesn't point to the list head;
        *  Instead of head of the list it contains address of 'p'


        /* And here is the problem number 2:
         * Even if we change declaration
         *         packet **i = &pending_list;
         * to
         *         packet *i = pending_list;
         * and then scroll the 'i' so that it points to 'p'
         * Now we just set p to p->next and next command deallocates the 'p' - so the pointer
         * which pointed to to 'p' points to a junk
         * See figures on the next page */

        if (*i != NULL) {
                (*i) = (*i)->next;
        }


        if (p != NULL) {
                packetfree(p);
        }
}
```

Sebastian Mountaniol for Tonian

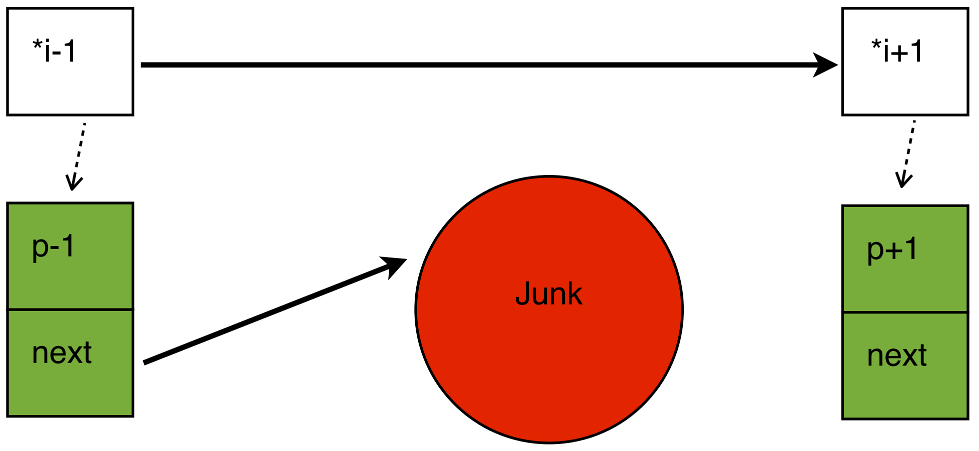
**1. Snippet 3: Before deleting a node.**



**2. *i = (*i)->next**



**3. free(p)**


*i-1
*i+1
p-1
next
Junk
p+1
next