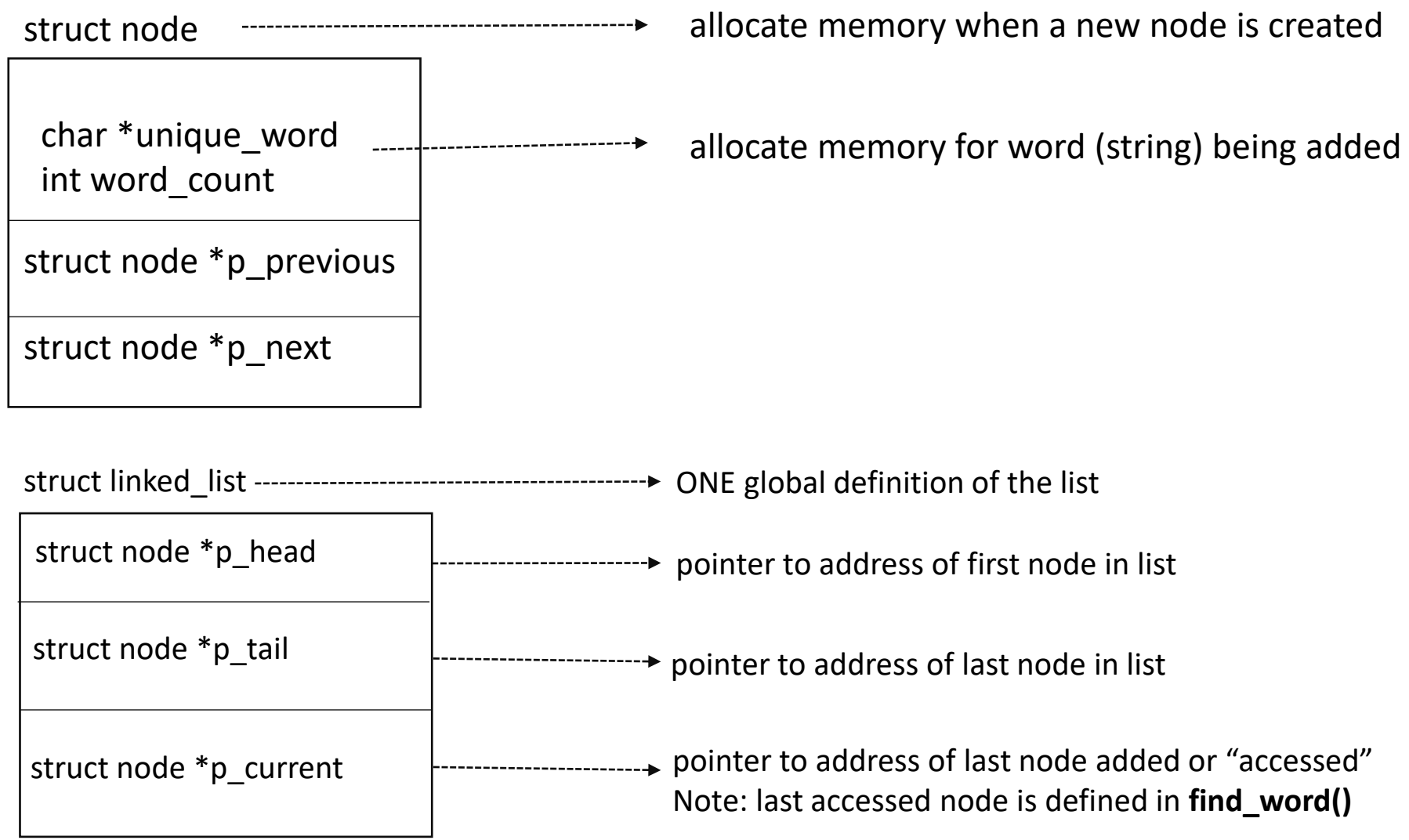
Structures used to maintain doubly-linked list



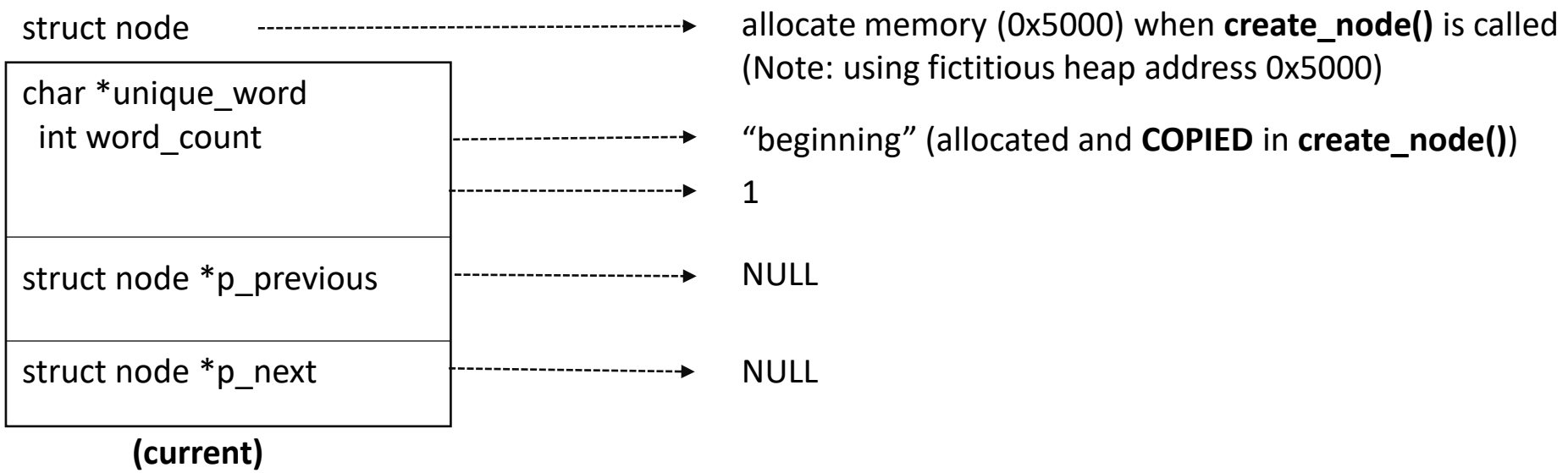
The following (not complete) test program is used for the linked list illustration that follows. Note we have added a utility function print_list() to output the word contents of the linked list to help with testing.

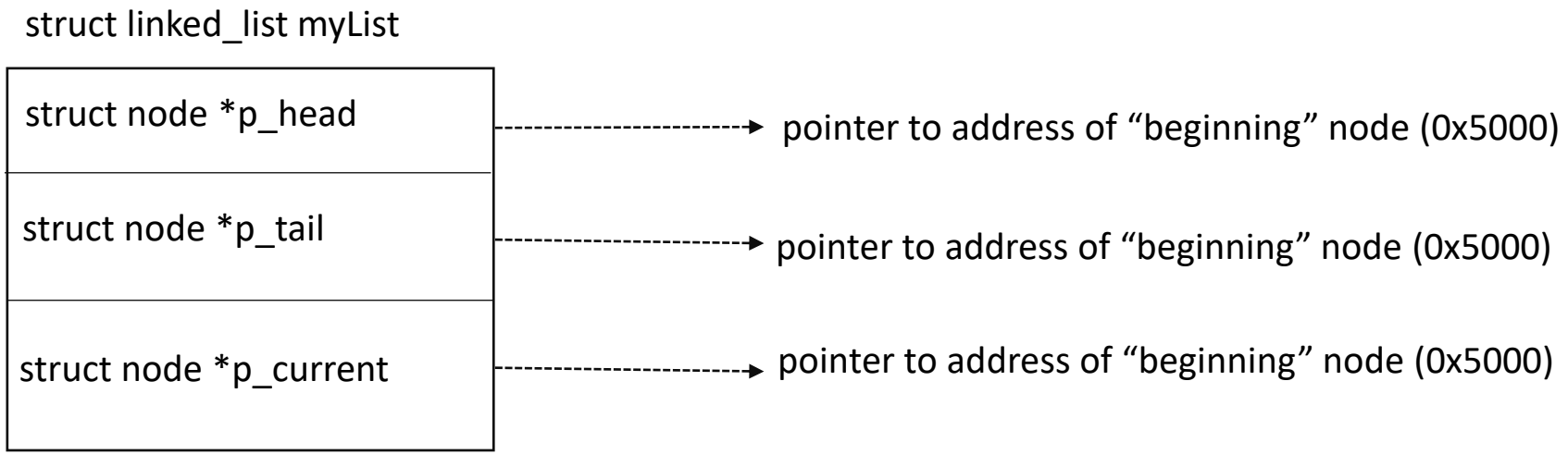
```
print_list(struct linked_list *p_list) {
        struct node *current = p_list->p_head;
        while( current != NULL){
            printf("%s  ", current->unique_word);
            current = current->p_next;
        }
        printf("\n");
        return;
}

main() {
        struct linked_list myList ;
        memset( &myList, 0, sizeof( myList ) ) ;
        add_node_at_head( &myList, "beginning" ) ;
        print_list(&myList);
        add_node_after_current( &myList, "middle");
        print_list(&myList);
        add_node_after_current( &myList, "whatever");
        print_list(&myList);
        find_word(&myList, "sponge");
        add_node_after_current( &myList, "sponge");
        print_list(&myList);
}
```

Sample run

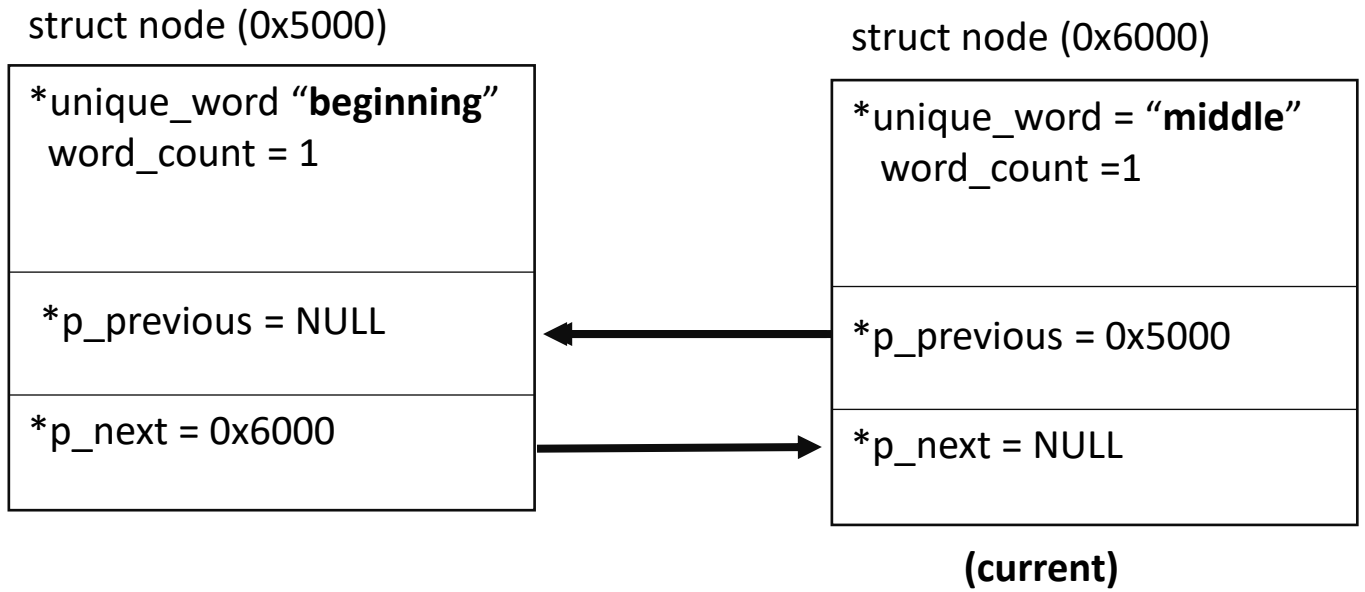
```
bash-4.4$ ./test
beginning
beginning  middle
beginning  middle  whatever
beginning  middle  sponge  whatever
bash-4.4$
```

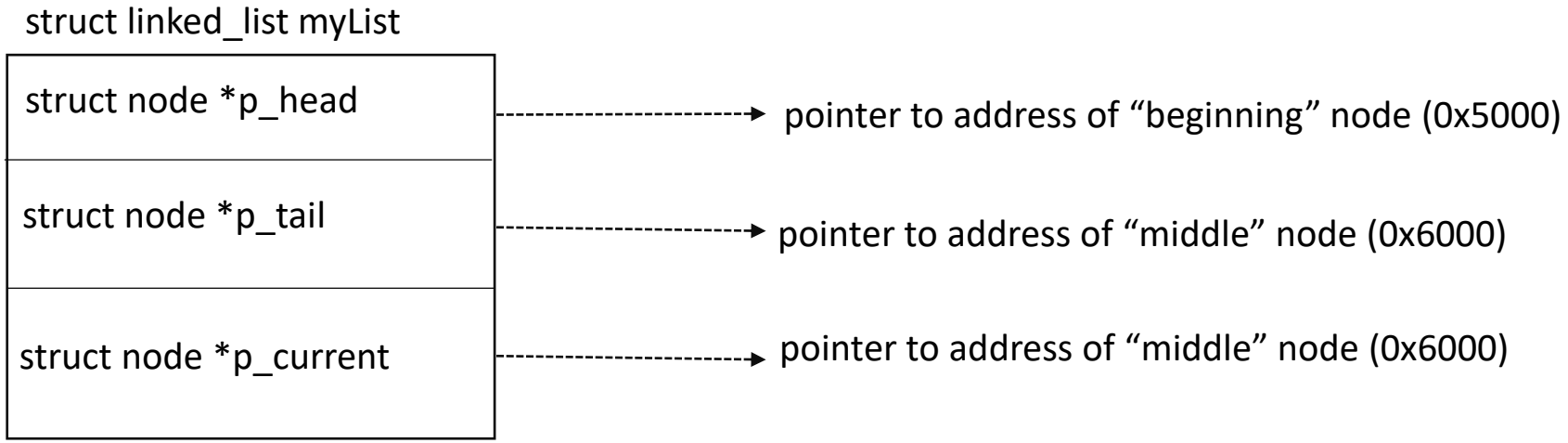
```
struct linked_list myList;                    // all fields initialized to NULL via memset()
add_node_at_head( &myList, "beginning")       // add_node_at_head() calls create_node()
```



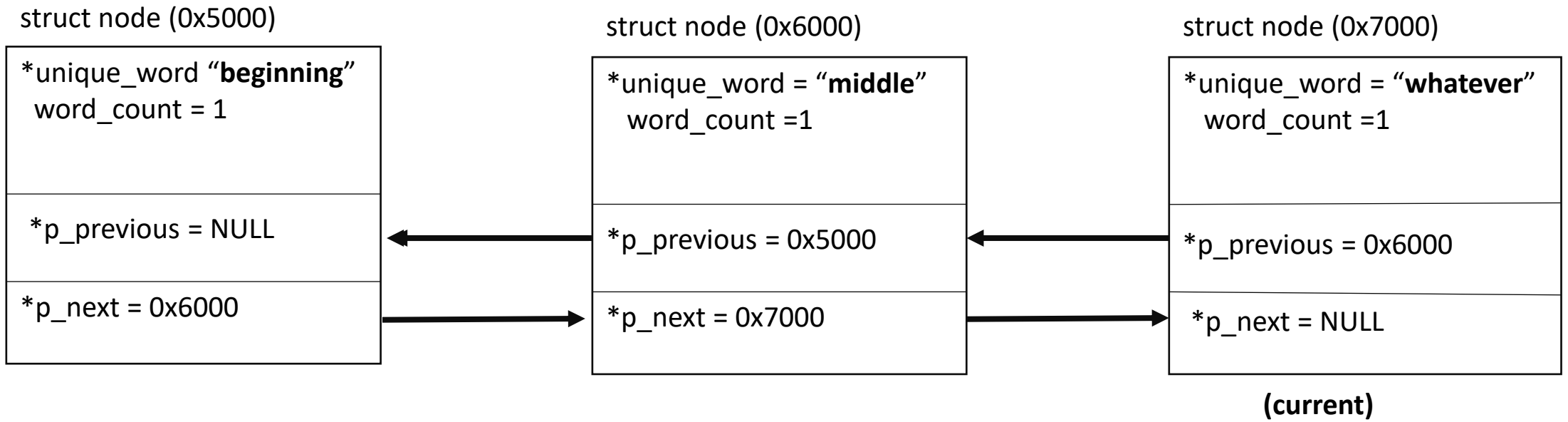


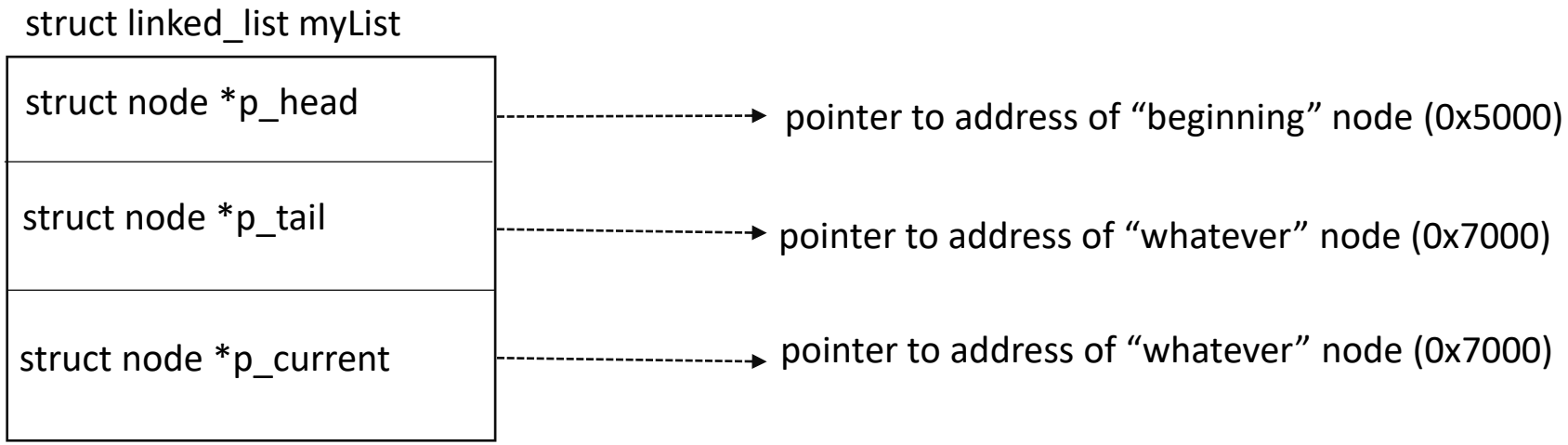
```
add_node_after_current( &myList, "middle") // add_node_after_current() calls create_node()
```



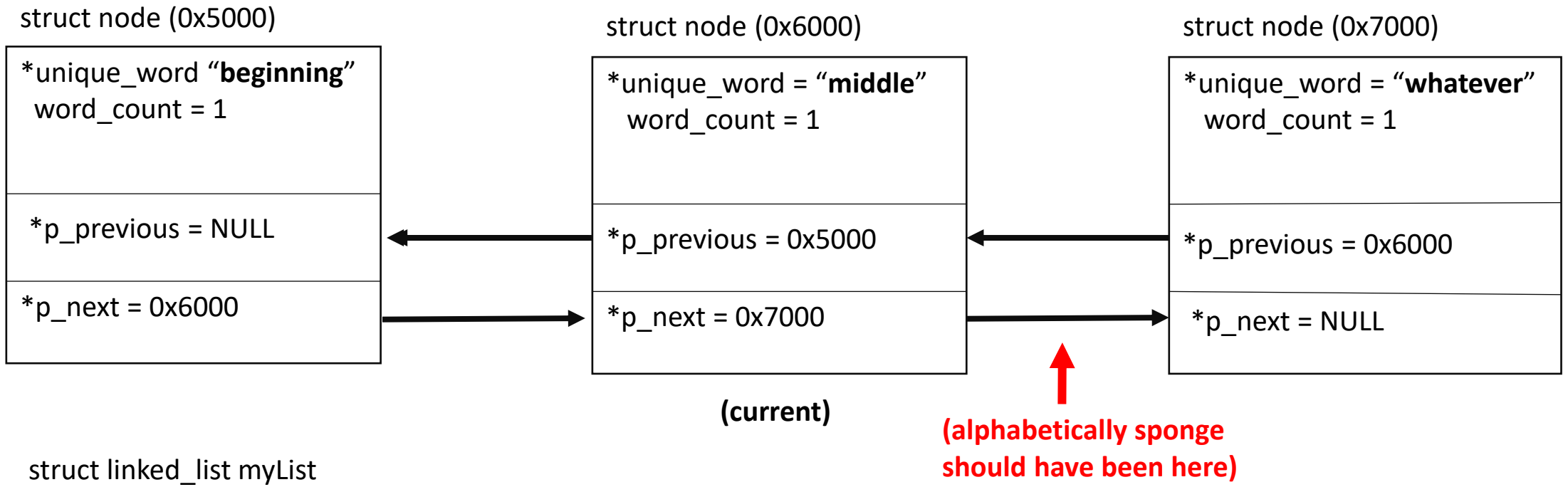


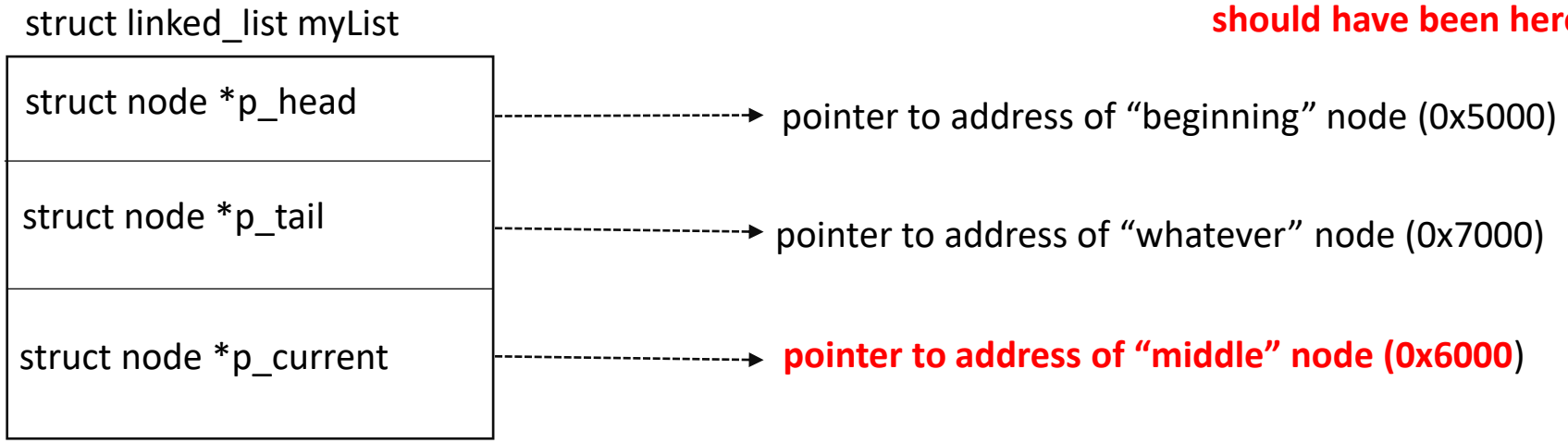
```
add_node_after_current( &myList, "whatever")    // add_node_after_current() calls create_node()
```





```
find_word( &myList, "sponge")      // "sponge" is not in the list, returns -1 (not success)
                                   // current will be set to "middle" node
                                   // list remains unchanged except for current pointer
```





```
add_node_after_current( &myList, "sponge")      // relies on current being set correctly!
                                                // note re-wiring of pointers
```