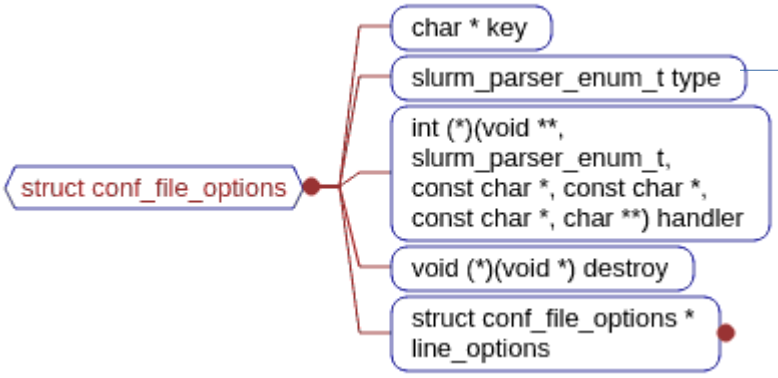
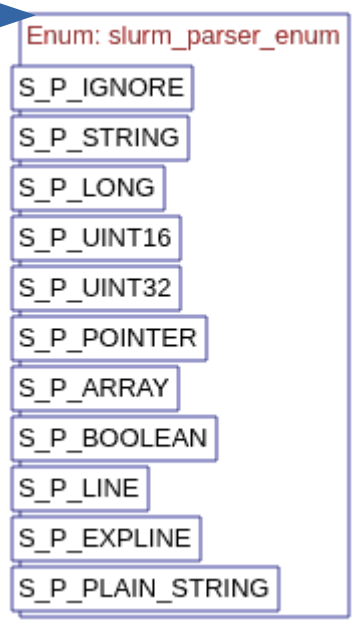
Layout structure details

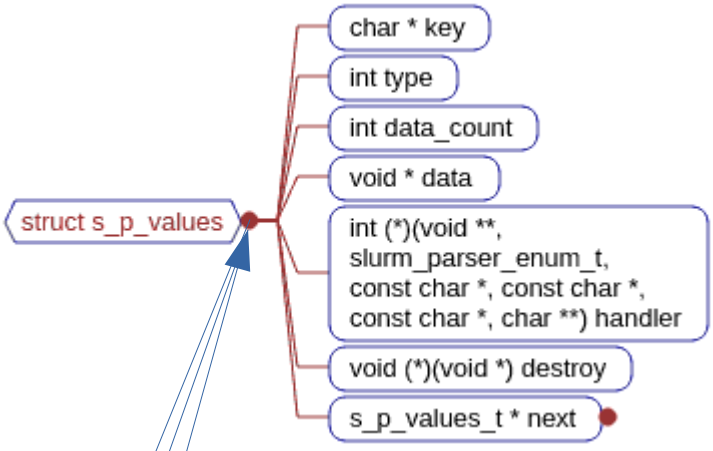
---

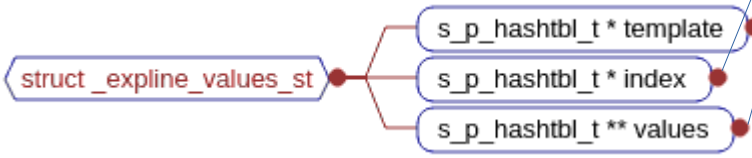
1) parse_config.[h&c]

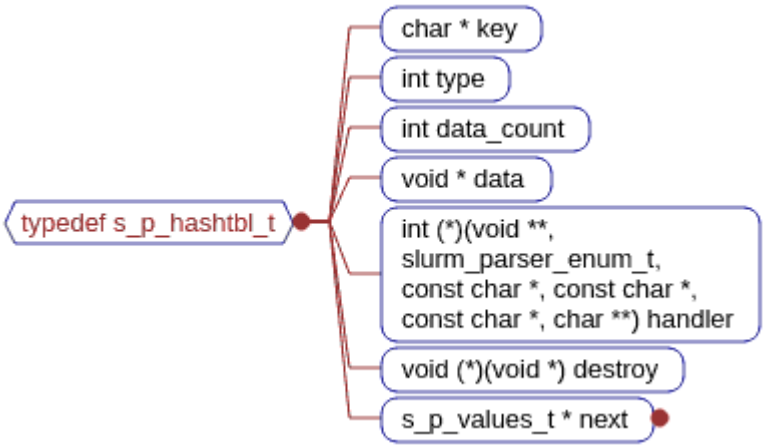


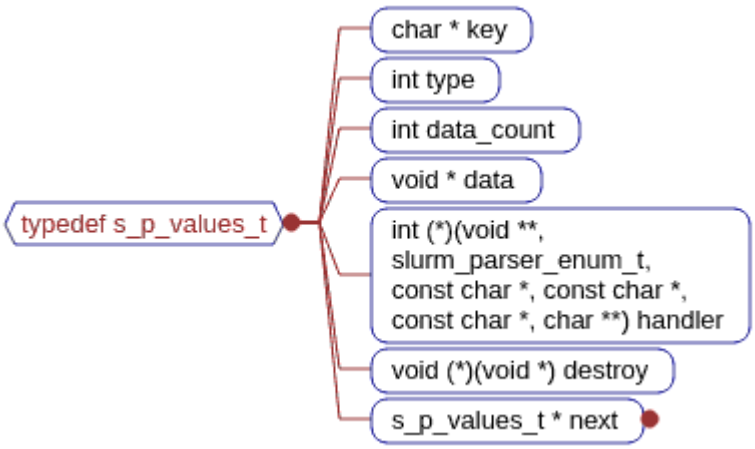



```
typedef struct slurm_parser_enum slurm_parser_enum_t;
typedef struct s_p_values s_p_values_t;
typedef s_p_values_t * s_p_hashtbl_t;
typedef struct conf_file_options *s_p_options_t;
```



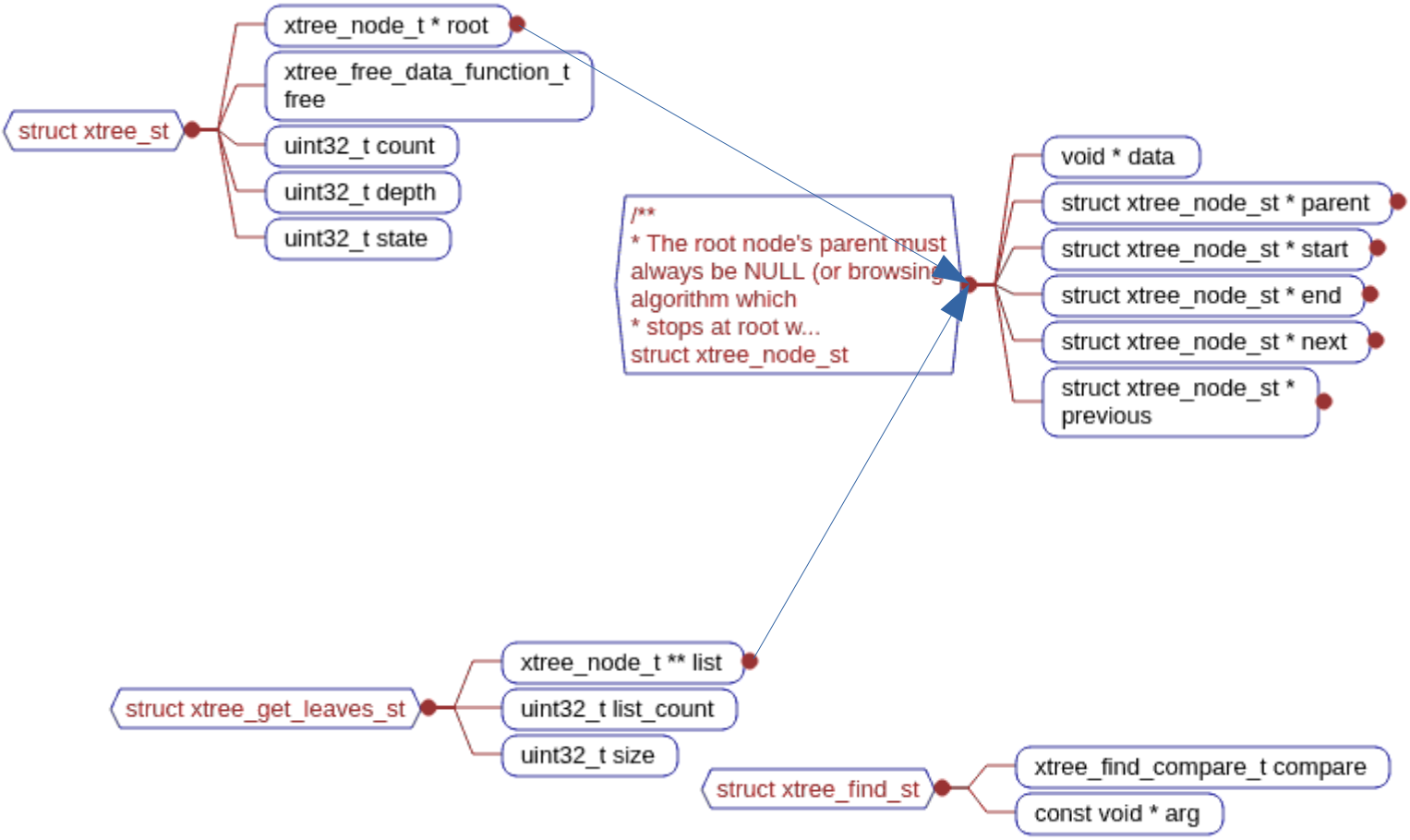






2) xtree.[h&c]

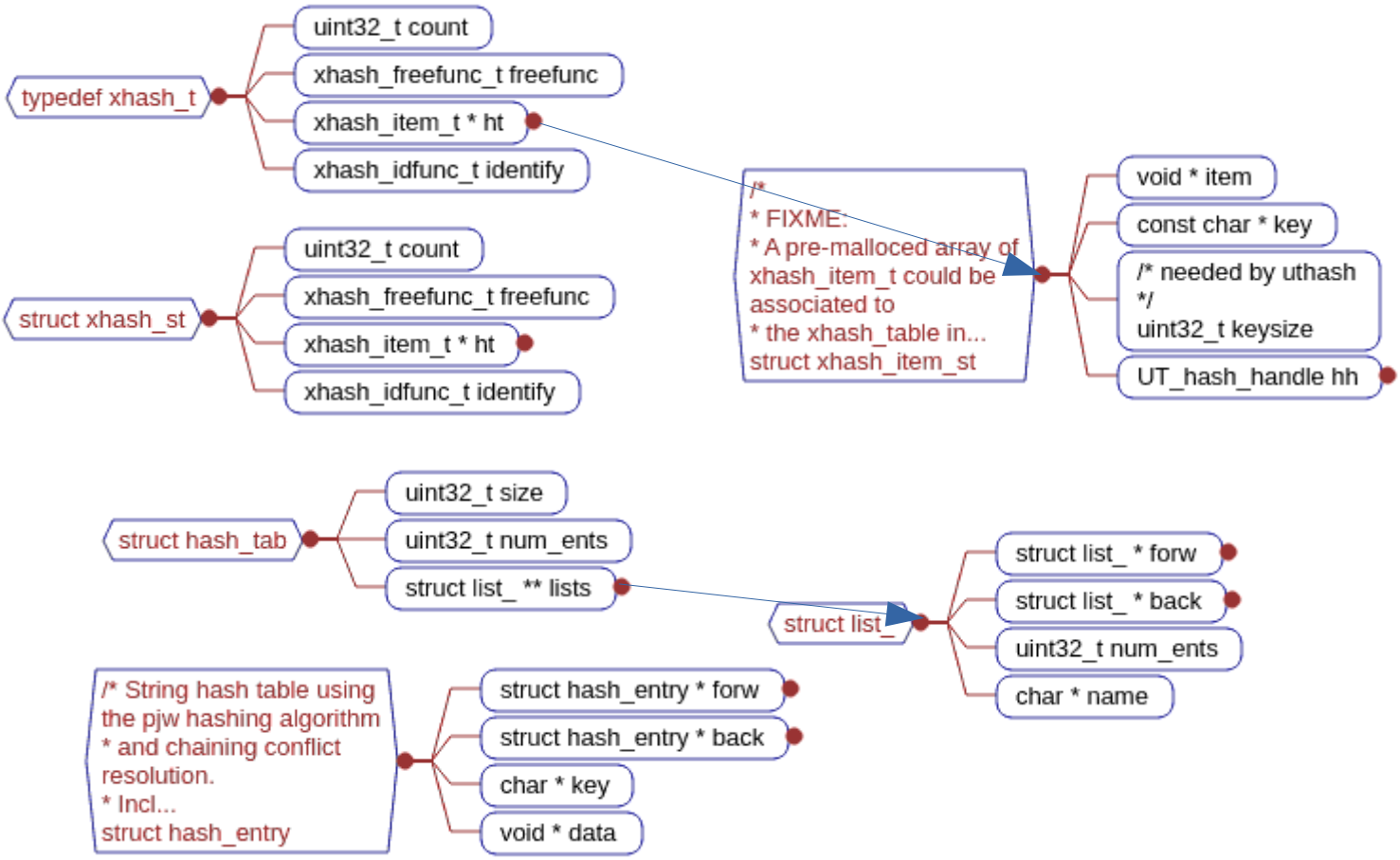
```
typedef struct xtree_node_st xtree_node_t;

typedef struct xtree_st xtree_t;
```



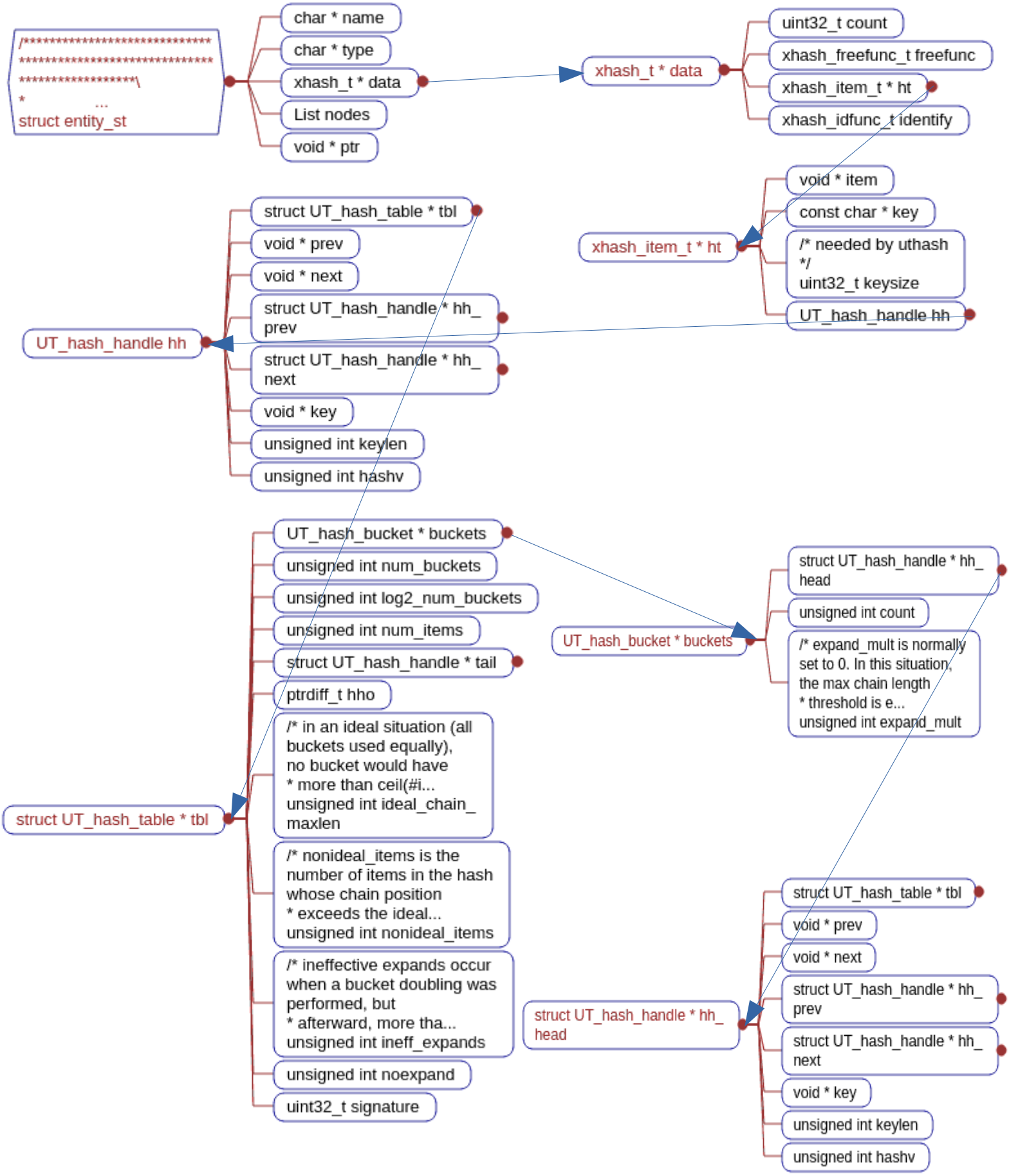

```
/** see function prototype for xtree_walk for description */

#define XTREE_PREORDER 1 #define XTREE_INORDER 2 #define XTREE_ENDORDER 4

#define XTREE_LEAF 8 #define XTREE_GROWING 16 /*XTREE_LEAF indicates the node
being visited is a leaf and receive consequently only one visit.

* XTREE_GROWING is called before any other calls, allowing a node to add childs.
*/

#define XTREE_LEVEL_MAX UINT32_MAX

#define XTREE_STATE_DEPTHCACHED 1

#define XTREE_PREPEND 1 /** prepend to child list */

#define XTREE_APPEND 2  /** append to child list */

#define XTREE_REFRESH_DEPTH 4 /** default: don't refresh at insertion */

#define xtree_node_get_data(node) ((node) ? (node)->data : NULL)

#define xtree_get_root(tree) ((tree) ? (tree)->root : NULL)

#define XTREE_GET_PARENTS_FIRST_SIZE 64
```

3) xhash.[h&c]

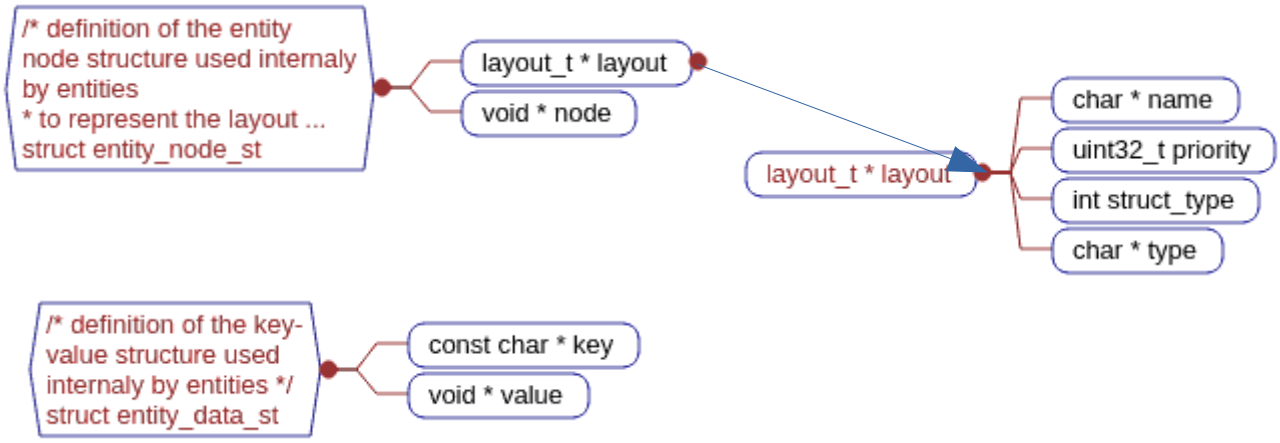
```
typedef struct xhash_st xhash_t;

typedef struct xhash_item_st xhash_item_t;

#if 0 /* undefine default allocators */

#undef uthash_malloc #undef uthash_free/* re-define them using slurm's ones */
#define uthash_malloc(sz) xmalloc(sz) #define uthash_free(ptr, sz) xfree(ptr)
#endif
```



4) entity.h

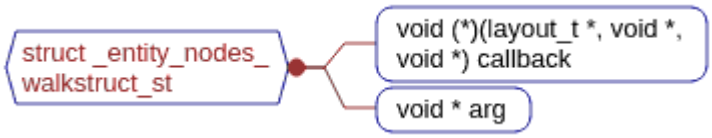

/*****************************
****
******************\
*
...
struct entity_st
char * name
char * type
xhash_t * data
List nodes
void * ptr
xhash_t * data
uint32_t count
xhash_freefunc_t freefunc
xhash_item_t * ht
xhash_idfunc_t identify
xhash_item_t * ht
void * item
const char * key
/* needed by uthash */
uint32_t keysize
UT_hash_handle hh
UT_hash_handle hh
struct UT_hash_table * tbl
void * prev
void * next
struct UT_hash_handle * hh_prev
struct UT_hash_handle * hh_next
void * key
unsigned int keylen
unsigned int hashv
struct UT_hash_table * tbl
UT_hash_bucket * buckets
unsigned int num_buckets
unsigned int log2_num_buckets
unsigned int num_items
struct UT_hash_handle * tail
ptrdiff_t hho
/* in an ideal situation (all buckets used equally), no bucket would have * more than ceil(#i... unsigned int ideal_chain_maxlen
/* nonideal_items is the number of items in the hash whose chain position * exceeds the ideal... unsigned int nonideal_items
/* ineffective expands occur when a bucket doubling was performed, but * afterward, more tha... unsigned int ineff_expands
unsigned int noexpand
uint32_t signature
UT_hash_bucket * buckets
struct UT_hash_handle * hh_head
unsigned int count
/* expand_mult is normally set to 0. In this situation, the max chain length * threshold is e... unsigned int expand_mult
struct UT_hash_handle * hh_head
struct UT_hash_table * tbl
void * prev
void * next
struct UT_hash_handle * hh_prev
struct UT_hash_handle * hh_next
void * key
unsigned int keylen
unsigned int hashv

```
/* definition of the entity itself, main structure of this file. */
typedef struct entity_st  entity_t;
/* definition of the key-value structure used internaly by entities */
typedef struct entity_data_st  entity_data_t;
/* definition of the entity node structure used internaly by entities * to
represent the layout nodes that are linked to them */
typedef struct entity_node_st entity_node_t;
```



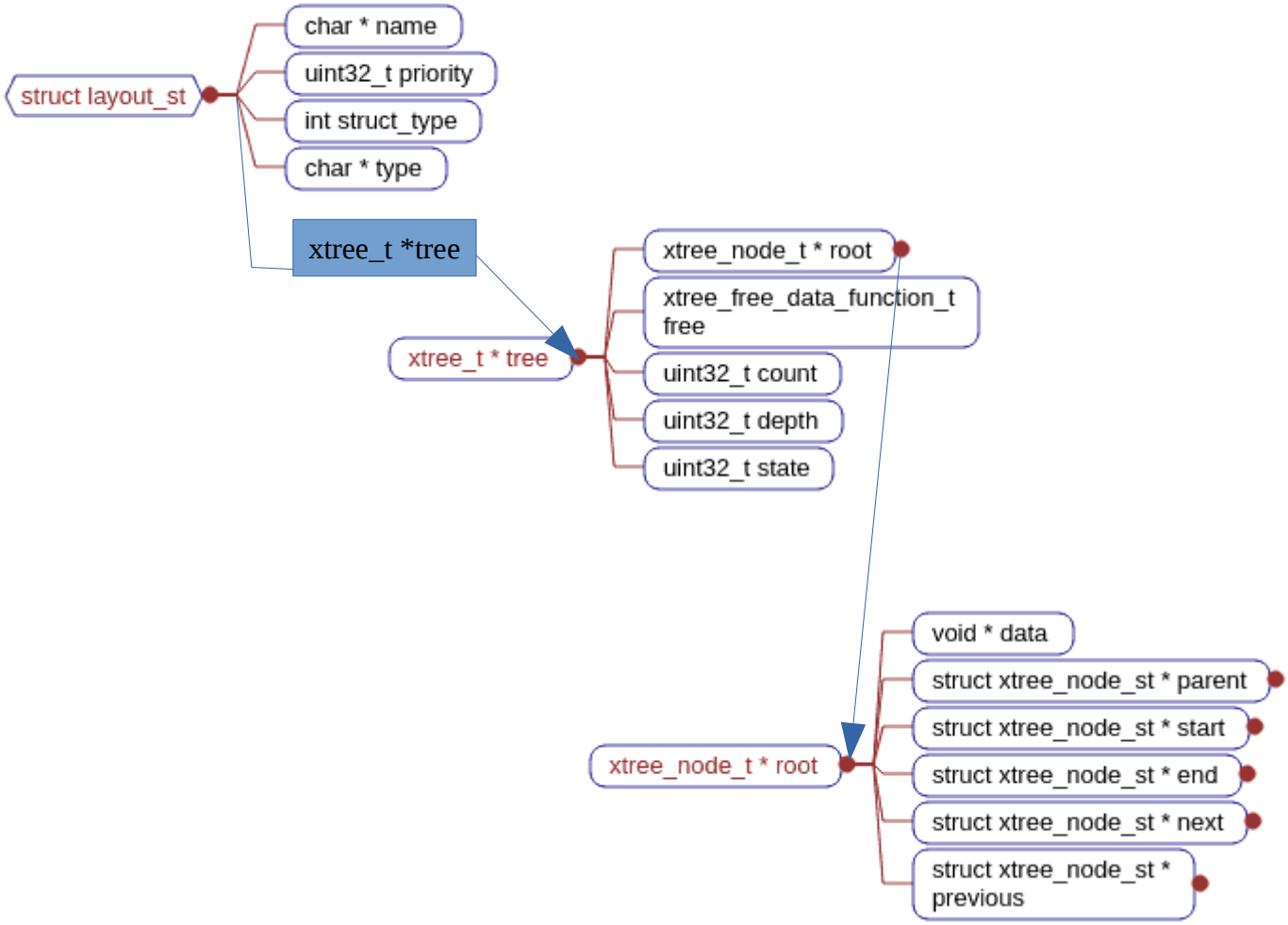
5) entity.c

```
typedef struct _entity_nodes_walkstruct_st _entity_nodes_walkstruct_t;
```



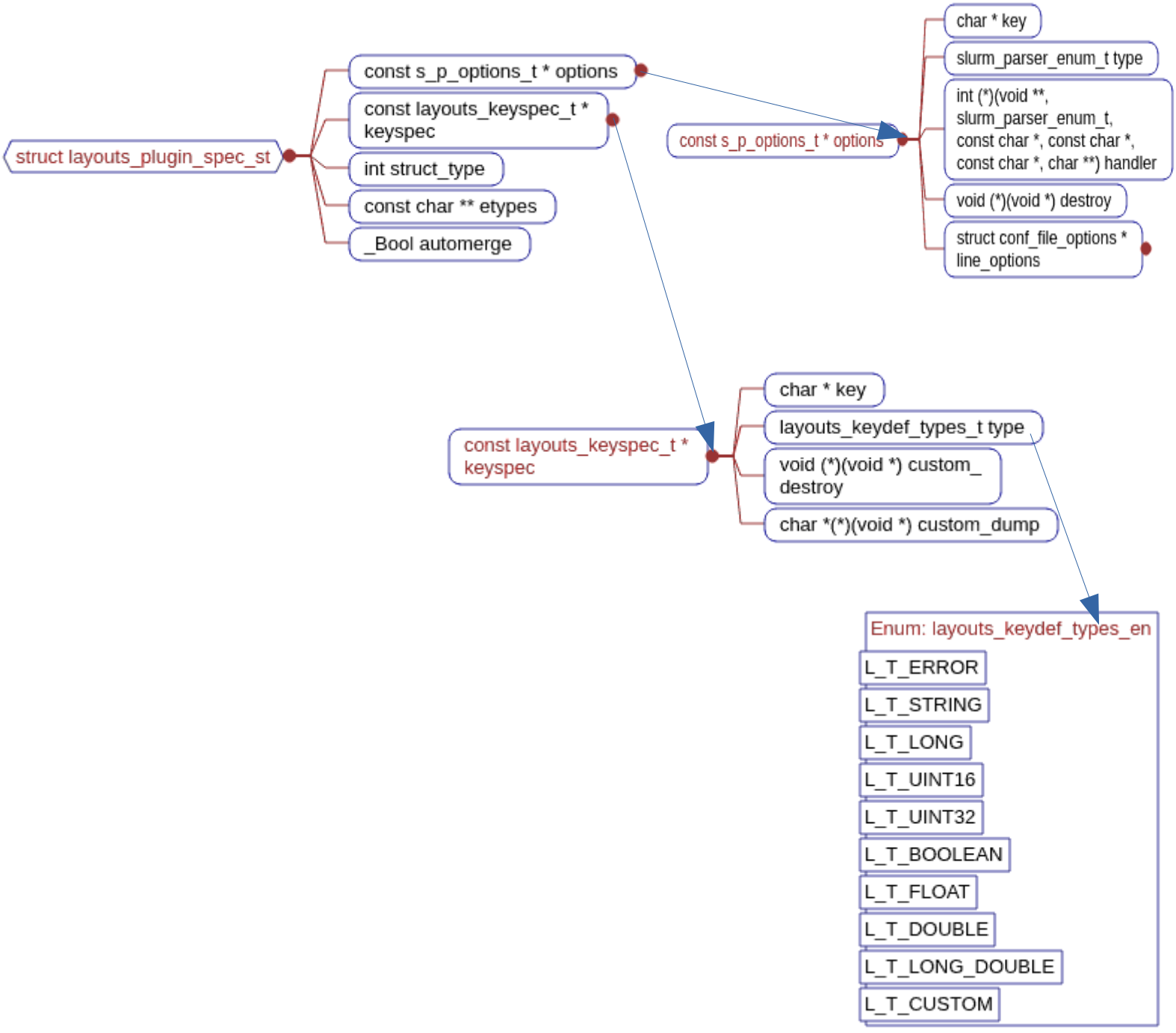
6)layout.h

```
typedef struct layout_st  layout_t;

#define LAYOUT_STRUCT_TREE 1
```



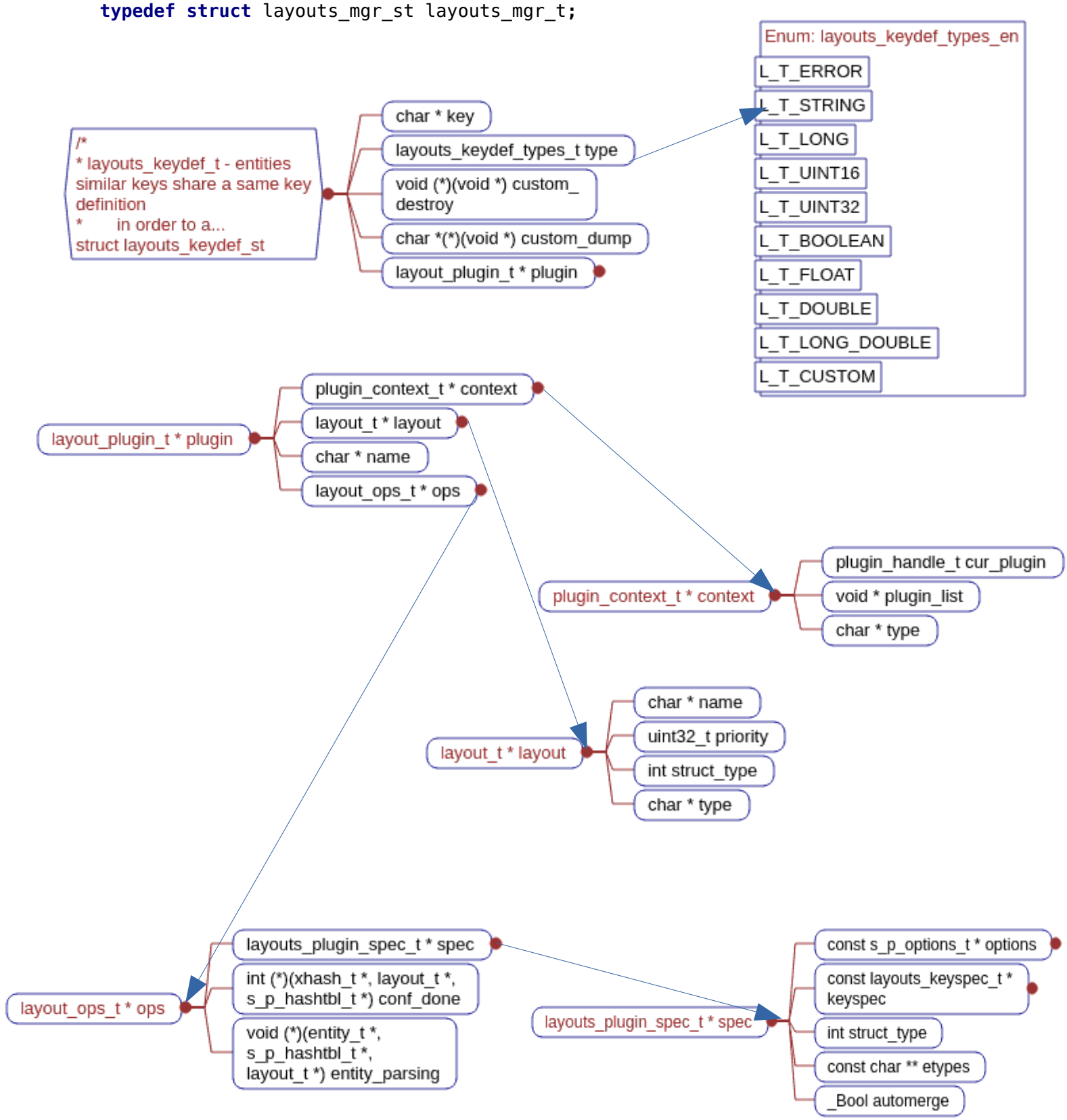
7)layouts_mgr.h

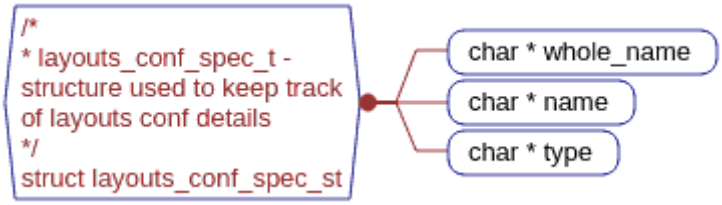
```
typedef enum layouts_keydef_types_en layouts_keydef_types_t;

typedef struct layouts_keyspec_st  layouts_keyspec_t;

typedef struct layouts_plugin_spec_st  layouts_plugin_spec_t;
```

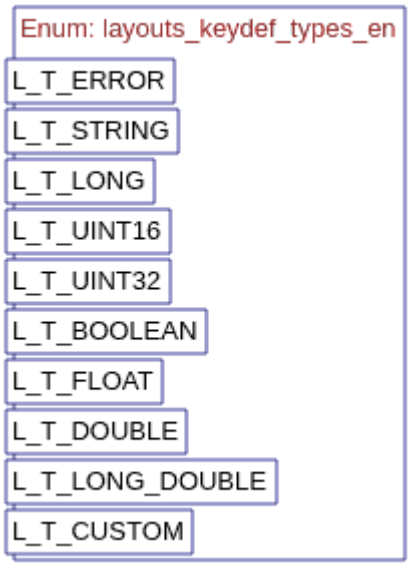


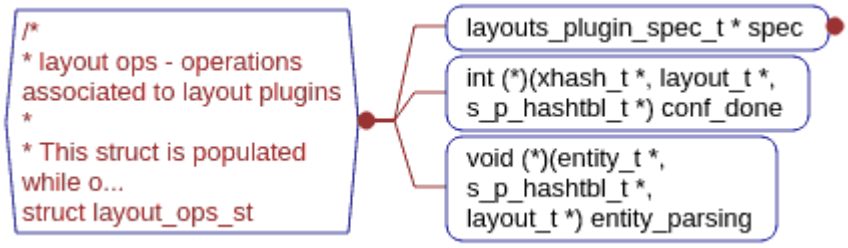
8)Layout_mgr.c

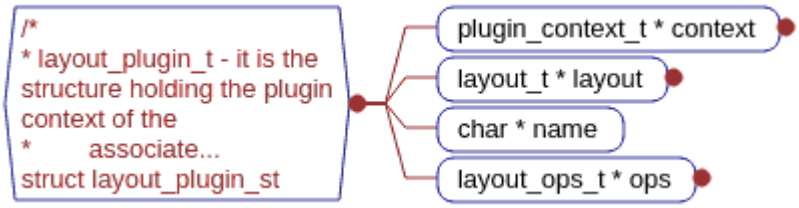
```
typedef struct layout_ops_st layout_ops_t;

typedef struct layouts_conf_spec_st layouts_conf_spec_t;

typedef struct layout_plugin_st layout_plugin_t;

typedef struct layouts_keydef_st layouts_keydef_t;

typedef struct layouts_mgr_st layouts_mgr_t;
```












```c
#define PATHLEN 256

void free(void*);

/* * layout plugin symbols - must be synchronized with ops structure definition
* as previously detailed, that's why though being a global constant, * it is
placed in this section. */

 const char *layout_syms[] = {

"plugin_spec", /* holds constants, definitions, ... */

"layouts_p_conf_done", /* */

"layouts_p_entity_parsing", };
```