LightNode SRM Information (eli5 at the bottom)

At offset 0x264 in En_Elf, there exists a pointer to a LightNode (pointing to 8011xxxx). When En_Elf is unloaded, the function `LightContext_RemoveLight` is run on the node:

```
void LightContext_RemoveLight(GlobalContext* globalCtx, LightContext* lightCtx, LightNode* node) {
	if (node != NULL) {
	if (node->prev != NULL) {
		node->prev->next = node->next;
	} else {
		lightCtx->listHead = node->next;
	}

	if (node->next != NULL) {
		node->next->prev = node->prev;
	}

	Lights_FreeNode(node);
	}
}
```

And given the LightNode struct is:

```
typedef struct LightNode {
  /*0x0*/  LightInfo* Info;
  /*0x4*/  struct LightNode* prev;
  /*0x8*/  struct LightNode* next;
} LightNode;
```

This means when you remove a LightNode using `LightContext_RemoveLight`, you essentially perform the following operations:

- Get the address found at `node+0x4`, add `0x8` to it, and write the value at `node+0x8` to the address
- Get the address found at `node+0x8`, add `0x4` to it, and write the value at `node+0x4` to the address

If we SRM this pointer to point to filename, we can perform an arbitrary RAM write to anywhere in memory.

So for example, if `node` was `0x8011aca0` (word before filename on 1.2) and your filename was:

- Filename pt1 (addr `0x8011aca4`): `0x80700000`
- Filename pt2 (addr `0x8011aca8`): `0x80701000`

What would happen upon running `LightContext_RemoveLight` is:

- Get the address found at `node+0x4`, so get the value at `0x8011aca4`, `0x80700000` in this case.
- Add `0x8` to it, so now we're looking at `0x80700008`
- Write the value at `node+0x8` to the looked at address, so we write `0x80701000` to `0x80700008`
  THEN
- Get the address found at `node+0x8`, so get the value at `0x8011aca8`, `0x80701000` in this case.
- Add `0x4` to it, so now we're looking at `0x80701004`
- Write the value at `node+0x4` to the looked at address, so we write `0x80700000` to `0x80701004`

However, since it's possible to sometimes have VC (and Wii U VC) ignore invalid writes (depending on the state of the dynarec cache when the code is run, if it's been cached, it won't ignore the invalid writes, however if it's not currently cached, it'll ignore invalid writes allowing for single-sided arbitrary value writing), we can put whatever value we want into one side of the filename and have it written to the memory location designated by the other half. In the demo example, we wanted to enable the debug menu, this requires a write of `0x00010001` to the address `0x801D9694`. To translate that into a filename, we have to account for the offsets in the node traversing process. Looking again at the first half of the remove operation:

- Get the address found at `node+0x4`, add `0x8` to it, and write the value at `node+0x8` to the address

(assuming you SRM `node` to equal `0x8011aca0`)
By subtracting `0x8` from our target address and putting it in the first half of the filename, the above operation will write the value of the 2nd half of our filename to our target address. So our debug menu srm filename will be `801D968C 00010001` ie ラとザォ0101.

N64 requires both filename halves to be valid, aligned pointers for this to work successfully.

ELI5:
To write value XXXXXXXX to address 80YYYYYY, set your filenames to either

1) 80YYYYYY-0x8 XXXXXXXX
2) XXXXXXXX 80YYYYYY-0x4

For double valid pointers, both of the above will happen, so for 80XXXXXX 80YYYYYY,

- 80XXXXXX will get written to 80YYYYYY+0x4
- 80YYYYYY will get written to 80XXXXXX+0x8

LightNode SRM's functionality is different depending on what hardware it's performed on. MIPS consoles (N64/iQue) LightNode SRM only works if word-aligned valid pointers (or nulls) are used, however PowerPC consoles (Wii, WiiU, GC) allow the use of misaligned pointers which allows categories like Defeat Ganon SRM to work since the last byte in one of the filename halves is written to the first byte of an entrance table entry.

Heaps and Angles:
1.2 Heap setup: https://www.youtube.com/watch?v=Udk4ckbeucY
1.2 Filename angle 0xaca0: https://www.youtube.com/watch?v=vZA7u2XCsoE
Better 0xaca0 by tsundere: https://www.youtube.com/watch?v=PgrgJS1xhWY

GC Heap setup: https://www.youtube.com/watch?v=RDb2cZBqBQM
MQ-Vanilla 0xb188 setup: start at 0x8000, 175 ess right to 0xb188 (omegalul)