Cómo crear tu propio RAM tester para Nintendo 64

Si estás empezando con **Libdragon** y quieres pasar del clásico «Hello World» a algo que interactúe de verdad con el hardware, aquí tienes un proyecto sencillo: un testador de memoria que detecta si tienes el **Expansion Pak** (8MB) o el **Jumper Pak** (4MB) y realiza pruebas de escritura/lectura.

1. El Makefile (Configuración del Proyecto)

Sustituye el contenido de tu Makefile por este. Está configurado para organizar los archivos y generar el archivo .z64 correctamente.

Makefile

```
V=1
SOURCE_DIR=src
BUILD_DIR=build
PROG_NAME=dashboard

include $(N64_INST)/include/n64.mk

all: $(PROG_NAME).z64

OBJS = $(BUILD_DIR)/main.o

$(PROG_NAME).z64: N64_ROM_TITLE="N64 RAM TESTER"

$(BUILD_DIR)/$(PROG_NAME).elf: $(OBJS)

clean:
	rm -rf $(BUILD_DIR) *.z64 *.elf

.PHONY: all clean
```

```
-include $(wildcard $(BUILD_DIR)/*.d)
```

---

2. Organiza tu espacio de trabajo

El Makefile ahora busca el código en una carpeta llamada src. Si tienes tu archivo en la raíz, muévelo con estos comandos:

Bash

```
mkdir -p src
mv main.c src/main.c
```

---

3. El Código: RAM & Expansion Test

Copia este código en src/main.c. Este programa detecta la capacidad de la consola y permite escribir el valor «1996» en una dirección física de la memoria para luego rescatarlo.

C

```
#include <stdio.h>
#include <libdragon.h>
#include <malloc.h>
#include <stdint.h>

int main(void) {
    timer_init();
      display_init(RESOLUTION_320x240, DEPTH_16_BPP, 2,
GAMMA_NONE, ANTIALIAS_RESAMPLE);
    joypad_init();

    int *puntero_secreto = NULL;
    int valor_recuperado = 0;
    uint32_t addr_fisica = 0;
    bool dato_en_memoria = false;
```

```
    while (1) {
        joypad_poll();
                                joypad_buttons_t    btn    =
joypad_get_buttons_pressed(JOYPAD_PORT_1);

        surface_t *disp = display_get();
        graphics_fill_screen(disp, 0);

          graphics_draw_text(disp, 20, 20, "== N64 RAM &
EXPANSION TEST ==");
        // Detección de memoria
        int mb = get_memory_size() / (1024 * 1024);
        char txt_cap[64];
        sprintf(txt_cap, "RAM Detectada: %d MB", mb);
        graphics_draw_text(disp, 40, 50, txt_cap);

        if (mb == 8) {
              graphics_draw_text(disp, 40, 65, "Hardware:
EXPANSION PAK OK");
        } else {
            graphics_draw_text(disp, 40, 65, "Hardware: JUMPER
PAK (4MB)");
        }

        // Acción [A]: Escribir en RAM
        if (btn.a) {
                if (!puntero_secreto) puntero_secreto =
(int*)malloc(sizeof(int));
            *puntero_secreto = 1996;
            addr_fisica = (uint32_t)puntero_secreto;
            dato_en_memoria = true;
            valor_recuperado = 0;
             graphics_draw_text(disp, 20, 150, "ESCRIBIENDO EN
RAM...");
            display_show(disp);
            wait_ms(500);
            disp = display_get();
        }

        // Acción [B]: Rescatar dato
        if (btn.b && dato_en_memoria) {
```

```
            valor_recuperado = *puntero_secreto;
        }

        if (dato_en_memoria) {
            char txt_addr[64];
               sprintf(txt_addr, "Dir: 0x%08X", (unsigned
int)addr_fisica);
            graphics_draw_text(disp, 40, 110, txt_addr);
        }

        if (valor_recuperado != 0) {
            char txt_res[64];
               sprintf(txt_res, "RECUPERADO: %d (EXITO)",
valor_recuperado);
            graphics_draw_text(disp, 40, 130, txt_res);
        }

        graphics_draw_text(disp, 20, 190, "[A] Escribir en RAM
[B] Rescatar");
        display_show(disp);
    }
    return 0;
}
```

---

4. Compilación Final

Para evitar errores de caché o archivos antiguos de otros ejemplos, usa siempre una compilación limpia:

Bash

```
libdragon make clean
libdragon make
```

¡Listo! Ya tienes tu archivo dashboard.z64 listo para probar en un emulador o en hardware real.