```rust
use std::{fs, process};
use std::fs::File;
use std::io::{self};
use std::io::{Write, Read, Error};
use std::time::{Instant, SystemTime};
use std::process::Command;
use chrono::{Utc, DateTime}; // MIT license
use device_query::{DeviceQuery, DeviceState, Keycode}; // MIT license

// This program calls lore-subprocess-capture-one-minute.exe
// The licence file covering both lore-rapid-fire-screenshots and lore-subprocess-capture-one-png
// will be written to \licenses each time lore-rapid-fire-screenshots.exe is run
// Log files are saved at \logs
// Screenshots are saved at \screenshots\rapid_fire_screenshots
// config\rapid_fire_screenshots\config.txt is used to read in the number of screenshots to take every 2-3 seconds
// If the file does not exist, it will be created with the default value of 30000

// IMPORTANT: The main monitor must remain on (not off) to avoid any errors that may occur at a later time
// otherwise the program will pause, but resume once the monitor is turned on again



fn main () {
    let start_time = Instant::now();
    let start_time_utc = Utc::now().time();

    const PROGRAM1: &str = "lore-rapid-fire-screenshots.exe";
    const VERSION: &str = "v1.0.2022";

    match fs::create_dir_all("./logs/") {
        Err(why) => println!("! {:?}", why.kind()),
        Ok(_) => {
            // nothing
        },
    }

    match fs::create_dir_all("./screenshots/rapid_fire_screenshots/") {
        Err(why) => println!("! {:?}", why.kind()),
        Ok(_) => {
```

```
            // nothing
        },
    }

    // log file
    let mut file = fs::OpenOptions::new()
        .read(true)
        .write(true)
        .create(true)
        .append(true)
        .open("./logs/log.txt")
        .unwrap();

    // START log file
    let system_time = SystemTime::now();
    let datetime: DateTime<Utc> = system_time.into();
    write!(file, "[{} UTC] START: {} ({}) running from {:?}\n", datetime.format("%Y-%m-%d %T"), PROGRAM1, VERSION, std::env::current_exe().unwrap());

    println!("{} {}", PROGRAM1, VERSION);
    println!();
    println!("-- Screenshots of your main display will be saved every 2-3 seconds,");
    println!("-- depending on your system, monitor resolution, and scenes.");
    println!("-- Edit 'config.txt' at '\\config\\rapid_fire_screenshots'");
    println!("-- and save the number of screenshots to take as a whole number.");
    println!("-- '30000' should be around 24 hours, depending on image details.");
    println!("-- If 'config.txt' does not exist, the number of screenshots will default to '30000'.");
    println!("-- They will be saved as png files at '\\screenshots\\rapid_fire_screenshots'\n");
    println!("-- Minimize this window while it captures screenshots as you work.");
    println!("-- Leave your main monitor on while the program is running to avoid errors.");
    println!("-- Start time UTC: {}\n", start_time_utc);

    let message = "Start time UTC: ".to_string() + &start_time_utc.to_string() + "\n";
    log_info(&message);

    write_license();

    let mut num_shots: u64 = 30000; // default value if configuration file is not found
    let config = load_config();
    if config.is_ok(){
        num_shots = config.unwrap().parse().unwrap()}
    else {
        create_config_file() // create config file with default value of 30000
    }

    println!("-- Number of screenshots to take: {}", &num_shots);
    println!("-- Press and hold down the 'END' key to quit early.");
    println!();

    println!("Capturing screenshots");

    let end_shot = num_shots + 1;
```

```rust
    let mut current_shot = 1;


    loop {

        if current_shot == end_shot {
            break
        }

        let output = if cfg!(target_os = "windows") {
            Command::new("cmd")
                .args(["/C", "lore-subprocess-capture-one-png"])
                .output()
                .expect("failed to execute process")
        } else {
            Command::new("sh")
                .arg("-c")
                .arg("echo failed to run program")
                .output()
                .expect("failed to execute process")
        };

        let mut exit_status = output.status.to_string();

        while exit_status.contains("1") {
            println!("error capturing screenshot, trying again");

            // try again
            let output = if cfg!(target_os = "windows") {
                Command::new("cmd")
                    .args(["/C", "lore-subprocess-capture-one-png"])
                    .output()
                    .expect("failed to execute process")
            } else {
                Command::new("sh")
                    .arg("-c")
                    .arg("echo failed to run program")
                    .output()
                    .expect("failed to execute process")
            };

            exit_status = output.status.to_string();
        }

        println!("Screenshot: {} -- Press and hold down 'END' to quit early", current_shot);

        // listen for the 'END' keypress from any active window to quit the program early
        let device_state = DeviceState::new();
        let keys: Vec<Keycode> = device_state.get_keys();
        if keys.contains(&Keycode::End) {
            end_program_by_keypress()
```

```rust
        }

        current_shot = current_shot + 1;
    }

    let end_time_utc = Utc::now().time();
    let duration = start_time.elapsed();
    println!("\n-- Done time (UTC): {}", end_time_utc);
    println!("-- Total time taken: {:?}\n", duration);

    let message = "Done time (UTC): ".to_string() + &end_time_utc.to_string() + "\n";
    log_info(&message);

    let system_time = SystemTime::now();
    let datetime: DateTime<Utc> = system_time.into();
    write!(file, "[{} UTC] INFO: {}", datetime.format("%Y-%m-%d %T"), "Total time taken: ");
    write!(file, "{:?}\n", duration);

    // END log file
    let system_time = SystemTime::now();
    let datetime: DateTime<Utc> = system_time.into();
    write!(file, "[{} UTC] END: {} ({}) \n", datetime.format("%Y-%m-%d %T"), PROGRAM1, VERSION);

}


fn log_info(message: &str) {

    let mut file = fs::OpenOptions::new()
    .read(true)
    .write(true)
    .create(true)
    .append(true)
    .open("./logs/log.txt")
    .unwrap();

    let system_time = SystemTime::now();
    let datetime: DateTime<Utc> = system_time.into();
    write!(file, "[{} UTC] INFO: {}", datetime.format("%Y-%m-%d %T"), message);
}


fn load_config() -> Result<String, Error> {

    let f = File::open("./config/rapid_fire_screenshots/config.txt");
    let mut f = match f {
        Ok(file) => file,
        Err(e) => return Err(e),
    }; // trap errors

    let mut text = String::new();
```

```rust

    f.read_to_string(&mut text)?;

    Ok(text)
}


fn end_program_by_keypress() {
    const PROGRAM1: &str = "lore-rapid-fire-screenshots.exe";
    const VERSION: &str = "v1.0.2022";

    let mut file = fs::OpenOptions::new()
    .read(true)
    .write(true)
    .create(true)
    .append(true)
    .open("./logs/log.txt")
    .unwrap();

    println!("\n'END' was pressed to quit early.\n");
    let message = "'END' was pressed to quit early.".to_string() + "\n";
    log_info(&message);

    let system_time = SystemTime::now();
    let datetime: DateTime<Utc> = system_time.into();
    write!(file, "[{} UTC] END: {} ({}) \n", datetime.format("%Y-%m-%d %T"), PROGRAM1, VERSION);

    process::exit(0); // quit program
}


fn write_license() {

    const PROGRAM1: &str = "lore-rapid-fire-screenshots.exe";
    const PROGRAM2: &str = "lore-subprocess-capture-one-png.exe";
    const VERSION: &str = "v1.0.2022";

    match fs::create_dir_all("./licenses/") {
        Err(why) => println!("! {:?}", why.kind()),
        Ok(_) => {
            // nothing
        },
    }

    let mut file = fs::OpenOptions::new()
    .read(true)
    .write(true)
    .create(true)
    .append(false)
    .open("./licenses/LICENSE-Lore-Rapid-Fire-Screenshots_and_Lore-Subprocess-Capture-One-PNG.txt")
    .unwrap();
```

```
256
257     // create LICENSE
258     write!(file, "{} {}\n", PROGRAM1, VERSION);
259     write!(file, "{} {}\n", PROGRAM2, VERSION);
260     write!(file, "Copyright 2022 Tarjin Rahman\n");
261
262     write!(file, "Licensed under the MIT License\n");
263     write!(file, "\n");
264     write!(file, "Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files\n");
265     write!(file, "(the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge,\n");
266     write!(file, "publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do\n");
267     write!(file, "so, subject to the following conditions:\n");
268     write!(file, "\n");
269     write!(file, "The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software\n");
270     write!(file, "THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF\n");
271     write!(file, "MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE\n");
272     write!(file, "FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION\n");
273     write!(file, "WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.\n");
274     write!(file, "_____\n");
275     write!(file, "\n");
276     write!(file, "The following third-party libraries were used in lore-rapid-fire-screenshots.exe: ");
277     write!(file, "\nchrono = '0.4.19' is licensed under the MIT license -- see https://crates.io/crates/chrono");
278     write!(file, "\ndevice_query = '0.2.8' is licensed under the MIT license -- see https://crates.io/crates/device_query/0.2.8");
279     write!(file, "\n\n");
280     write!(file, "The following third-party libraries were used in lore-subprocess-capture-one-png.exe: ");
281     write!(file, "\nchrono = '0.4.19' is licensed under the MIT license -- see https://crates.io/crates/chrono");
282     write!(file, "\nscrap = '0.5.0' is licensed under the MIT license -- see https://crates.io/crates/scrap");
283     write!(file, "\nrepng = '0.2.2' is licensed under the MIT license -- see https://crates.io/crates/repng");
284 }
285
286
287 fn create_config_file() {
288
289     const PROGRAM1: &str = "lore-rapid-fire-screenshots.exe";
290     const PROGRAM2: &str = "lore-subprocess-capture-one-png.exe";
291     const VERSION: &str = "v1.0.2022";
292
293     match fs::create_dir_all("./config/rapid_fire_screenshots") {
294         Err(why) => println!("! {:?}", why.kind()),
295         Ok(_) => {
296             // nothing
297         },
298     }
299
300     let mut file = fs::OpenOptions::new()
301     .read(true)
302     .write(true)
303     .create(true)
304     .append(false)
305     .open("./config/rapid_fire_screenshots/config.txt")
306     .unwrap();
```

```
307
308     // create config.txt with default value
309     write!(file, "30000");
310 }
```