Easter Bunny King Competition

Developer Documentation

Name: Haseeb Raza
Neptun Code: OT7DEE
Professor: Szabó Dávid
Course: Operating Systems Lab Assignment

May 5, 2025

Contents

1 Introduction

This developer documentation describes the "Easter Bunny King" competition, a C program simulating a playful contest where bunny boys recite poems to earn red eggs from bunny girls. The program demonstrates key Linux system programming concepts, including process creation (`fork`), inter-process communication (pipes), signal handling, and file I/O.

2 Program Features

- **Bunny Registration**: Add a bunny with name and poem (eggs start at 0).
- **Bunny Management**: Modify, delete, or list registered bunnies.
- **Competition Simulation**:
  - Fork a child process for each bunny.
  - Child sends a `SIGUSR1` signal to the Chief Bunny.
  - Child recites its poem and generates a random egg count (1–20).
  - Sends egg count to parent through pipe.
- **Winner Declaration**: Declare bunny with the highest egg count.

3 System Architecture

3.1 Data Storage

Data is stored in `bunny_data.txt` with format:

```
BunnyName|PoemText|EggCount
```

Example:

```
red bunny|this is very cute one|1
blue bunny|this is a small one|2
green bunny|this is very innocent one|19
```

3.2 Process Flow

1. Parent forks a child per bunny.
2. Each child sends `SIGUSR1` to parent.

3. Child prints poem and generates random egg count.
4. Sends egg count to parent via pipe.
5. Parent reads from pipe, updates data, and declares the King.

4 System Calls and Mechanisms

4.1 Process Creation

```
pid_t pid = fork();
if (pid == 0) {
    // child logic
} else {
    // parent logic
}
```

4.2 Signal Handling

```
void handler(int signum) {
    bunnyArrived = 1;
}
signal(SIGUSR1, handler);
kill(getppid(), SIGUSR1);
```

4.3 Pipe Communication

```
int pipefd[2];
pipe(pipefd);

if (fork() == 0) {
    // Child
    close(pipefd[0]);
    int eggs = (rand() % 20) + 1;
    write(pipefd[1], &eggs, sizeof(int));
    close(pipefd[1]);
    exit(0);
} else {
    // Parent
    close(pipefd[1]);
    int received;
    read(pipefd[0], &received, sizeof(int));
```

```
    close(pipefd[0]);
}
```

5 Error Handling

- File open failure:

```
    FILE *fp = fopen(filename, "r");
    if (!fp) {
        perror("File error");
        return;
    }
```

- Pipe or fork failure:

```
    if (pipe(pipefd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }
    if (fork() == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }
```

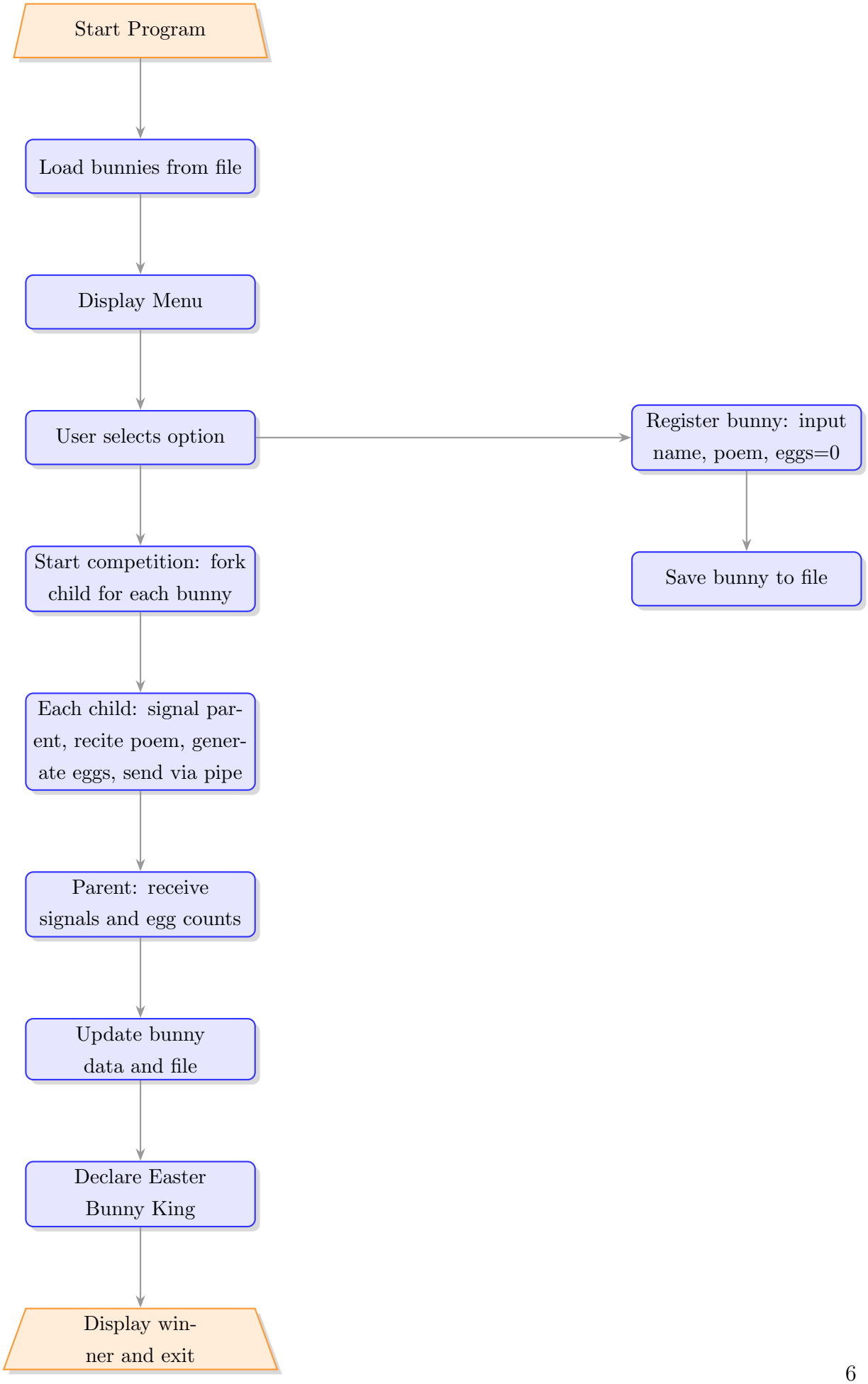
6 Compilation and Execution

6.1 Compilation

```
gcc -o bunny_check bunny_check.c
```

6.2 Execution

```
./bunny_check
```


Start Program
Load bunnies from file
Display Menu
User selects option
Register bunny: input name, poem, eggs=0
Save bunny to file
Start competition: fork child for each bunny
Each child: signal parent, recite poem, generate eggs, send via pipe
Parent: receive signals and egg counts
Update bunny data and file
Declare Easter Bunny King
Display winner and exit