OS2021: Challenge #2: Pokemon GamePlay Processes

Jordi Mateo Fornés `jordi.mateo@udl.cat`

University of Lleida (Campus Igualada -UdL) — **From:** October 29, 2020; **To**: November 26, 2020

Scope

A UNIX process or job is the result of executing a UNIX command. Processes are created by UNIX commands (including the commands that open windows in X), program executions (including GCC (1), mail (1) and programs you write and compile), and the C-shell command interpreter itself. At any moment, a process may be either running or stopped. The UNIX operating system provides many ways to control these processes, such as suspending, resuming and terminating.

Objectives

- To know the functions of creating fork and exec processes. Understand how they work, how they are called, what they return.
- To know the methods of synchronization between the exit and wait for processes and their relationship.
- To know pipes as communication mechanisms between processes.
- To work with signals between processes.
- To have fun!

Instructions

- This challenge can be delivered in groups between 2 or 3 students.
- The challenge must be delivered on GitHub and present in the Campus Virtual the link, with a summary of the tasks performed by each member.
- There are several different ways to approach these problems. It is your job to analyze them from an engineering point-of-view, determine the trade-offs, and to explain the implementation you select in the (GitHub repo).
- Do not underestimate the importance of the write-up. Your project grade depends significantly on how well you understood what you were doing, and the write-up is the best way for you to demonstrate that understanding.
- Create a READEME.md with the project information about author, workflow, design, implementation, files.
- Append to the README.md and answer the following questions: What have I learned from doing the challenge? How did I learn that? What has allowed me to improve? Why did it help me? Why can it serve me?

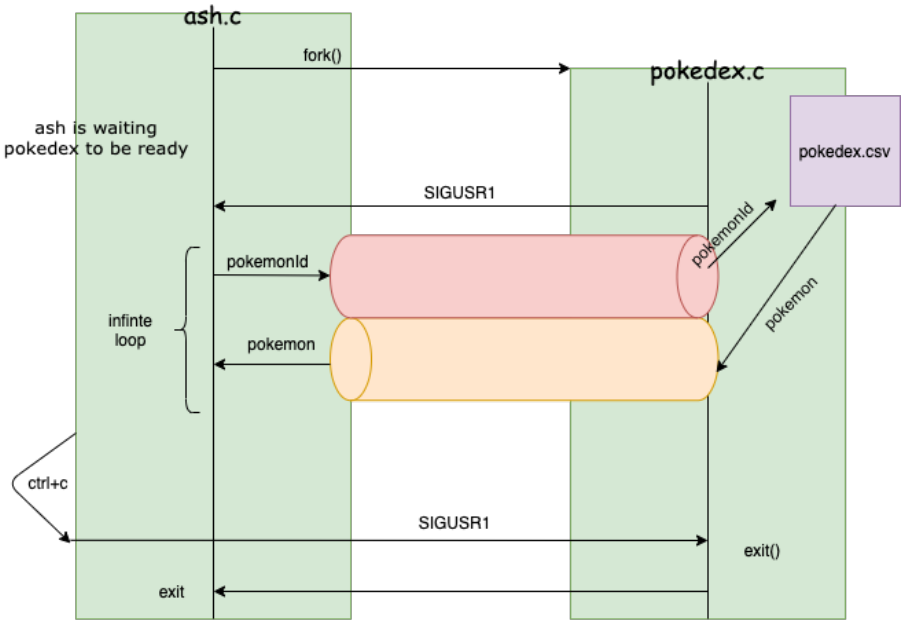
1 Checking information into the Pokedex — (2 points)

> I see now that one's birth is irrelevant. It's what you do that determines who you are.
> –Mewtwo

We want to make a process that simulates the activity of a Pokedex. To do it, we assume that Ash wants to check information related to pokemon at any time. We need to assume:

- We can not start working with the Pokedex until the device is ready. It will be ready when the process finishes reading into Memory the information of `pokedex.csv`.
- Pokedex is configured to read information from **stdin [0]** and writes information to **stdout [1]**.
- The file `pokedex.c` contains the code that read pokemon information from `pokedex.csv` and loads them into Memory.

So, `ash.c` will send the **pokemonId** that the user enters using the keyboard (**stdin [0]**), check that the **pokemonId** is between 1 and 151 (only first generation) and send this **pokemonId** through the pipe. When the Pokedex reads the **pokemonId**, it must return using another pipe the information related to the pokemon (the entire object) to Ash process. Note, that the responsibility to display pokemon information into **stdout [1]** belongs to Ash process, not Pokedex. These actions must be repeated in an infinite loop until Ash process receives (a shout for his mother -go to bed right now!) ctrl+c(SIGINT), then Ash must finish first the Pokedex (SIGUSR1) and once the Pokedex finishes, Ash could finish and go to sleep.



2 Pokemon capture adventure — (3 points)

> A Caterpie may change into a Butterfree, but the heart that beats inside remains the same
> –Brook

.

Let's go! Gotta Catch 'Em All! Now is time to simulate the gameplay of Pokemon Go (simplified in a very basic one xD). Ash process is going to create a child that will represent a wild Pokemon. The parent (Ash) must generate a random integer between 1 and 151 and send to the Pokedex to determine which Pokemon appears in front of us. The parent process shows a menu (**stdout [1]**) with the following actions, Throw Pokeball, Throw berry (optional) or run. Moreover, Ash is the responsible to write all the messages to **stdout [1]**.

```
jordi@jordi-VirtualBox:~/pokemon/capture$ ./ash
################
# E. Explore
# Q. Quit
################
E
Ash:[2657] --> Wild pokemon appeared [2659]
@@@@@@@@@@@@@@@@@@@@@@@@
 Farfetch'd (83)
 +++ Type1: Normal, Type2: Flying
 +++ Total: 352, Hp: 52,
 +++ Attack: 65, Defense: 55
 +++ SpAttack: 58, SpDefense: 62, Speed: 60
 +++ Gen: 1 Legendary: 0
@@@@@@@@@@@@@@@@@@@@@@@@
# P. Throw pokeball
# R. Run
P
Gotcha!The pokemon was caught.
################
# E. Explore
# Q. Quit
################
E
Ash:[2657] --> Wild pokemon appeared [2660]
@@@@@@@@@@@@@@@@@@@@@@@@
 Moltres (146)
 +++ Type1: Fire, Type2: Flying
 +++ Total: 580, Hp: 90,
 +++ Attack: 100, Defense: 90
 +++ SpAttack: 125, SpDefense: 85, Speed: 90
 +++ Gen: 1 Legendary: 1
@@@@@@@@@@@@@@@@@@@@@@@@
# P. Throw pokeball
# R. Run
P
Oh no!The pokemon broke free.
# P. Throw pokeball
# R. Run
P
Gotcha!The pokemon was caught.
################
# E. Explore
# Q. Quit
################
E
Ash:[2657] --> Wild pokemon appeared [2661]
@@@@@@@@@@@@@@@@@@@@@@@@
 Omanyte (138)
 +++ Type1: Rock, Type2: Water
 +++ Total: 355, Hp: 35,
 +++ Attack: 40, Defense: 100
 +++ SpAttack: 90, SpDefense: 55, Speed: 35
 +++ Gen: 1 Legendary: 0
@@@@@@@@@@@@@@@@@@@@@@@@
# P. Throw pokeball
# R. Run
P
Oh no!The pokemon broke free.
# P. Throw pokeball
# R. Run
P
The pokemon escaped already
################
# E. Explore
# Q. Quit
################
Q
!!!!I'm tired from all the fun...
jordi@jordi-VirtualBox:~/pokemon/capture$
```

1. **Throw pokeball**: The Pokemon uses a probability distribution to answer Ash action. We generate a random number between 1 and 10. If the number is 7, the Pokemon escaped and we lose the

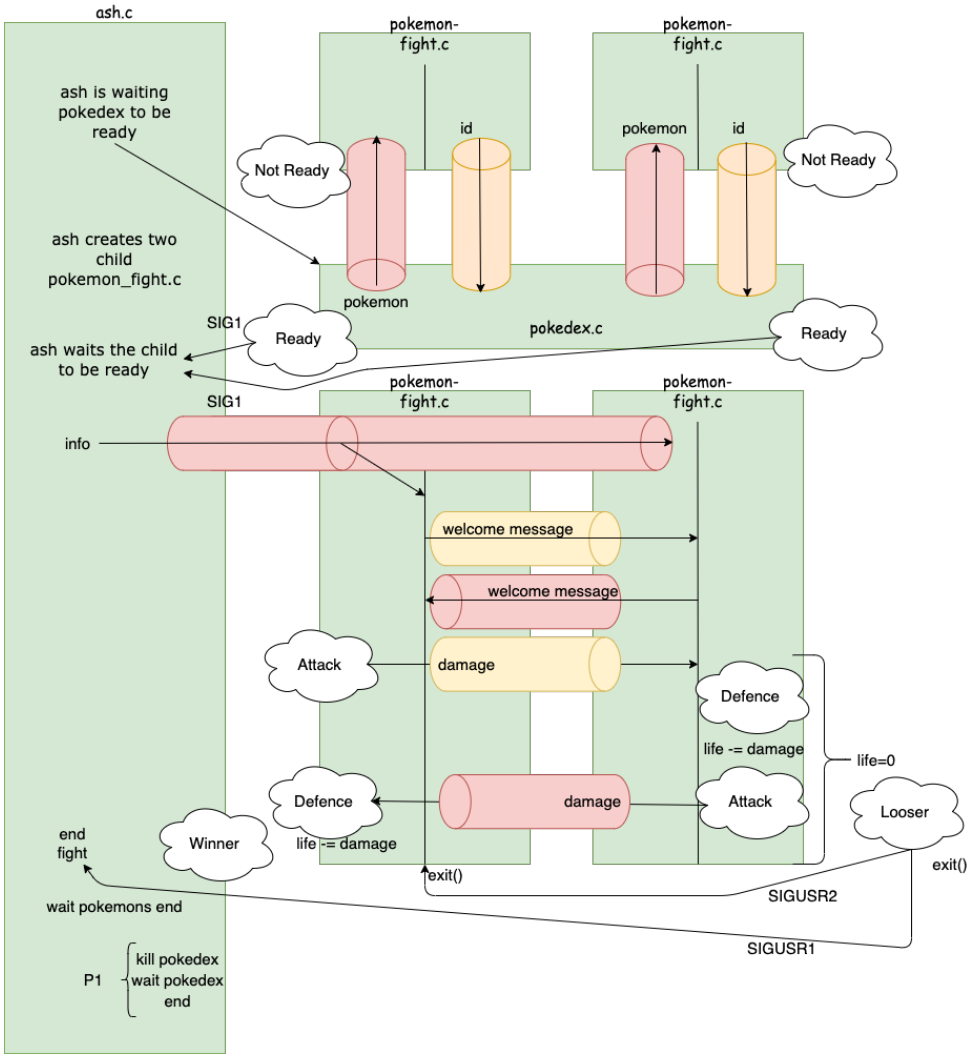
opportunity to catch (the child process Pokemon ends, after notifing the result to Ash process.). If the number is 2, then gotcha, Pokemon was caught. The others values represents that the Pokemon breaks free but we can try again (note here the Pokemon not ends).

2. **Throw a berry**:[Optional] Each berry increase the probability to catch the Pokemon. This way, the first berry allows a catch [2,4], second berry [2,4,6] and so on until max [2,4,6,8]. To simplify you can avoid the berries and assume that [2,4,6,8] allows Ash to catch the wild Pokemon.
3. **Run away**: Makes Pokemon process end, then Ash shows a certain log message (**stdout [1]**) and return to the main menu.

TIPS: The simplest solution uses signals and exit status code.

3 Simulating battles — (4 points)

Prepare for trouble! And make it double! To protect the world from devastation! To unite all peoples within our nation! To denounce the evils of truth and love! To extend our reach to the stars above! Jessie! James! Team Rocket blasts off at the speed of light! Surrender now, or prepare to fight!"



Now, it turns to fight with Team Rocket, and we need to implement a fighting simulator. We are going to simulate a battle between two Pokemon using turns (attack and defence). To make it easy, we are going to assume that each Pokemon generates a pseudo-random integer representing the damage of each turn. Both Pokemon are going to attack one time each other until one of those dies (life=0). So, each time a Pokemon receives one attack we must update their current life (currentlife=-damage). Each Pokemon process is responsible for printing in the **stdout [1]** the log of the battle and must use a different colour to display its text. The steps of the process Pokemon is:

- Generate a random integer between 1 and 151.
- Ask information about this Pokemon to the Pokedex process.
- Tell Ash (parent) that it is ready to fight.
- Wait to obtain information about their configuration (file descriptor, colours,etc.) [This step is optional, not required.]
- Welcoming the Pokemon rival exchanging information (PID and pokemon).
- Fight. **Note**. To decide which Pokemon starts you can pass as an argument in the exec, or with config information. Feel free to implement what you want.
- When the Pokemon attacks, it generates a random integer and sends to the other Pokemon.
- When the Pokemon defend waits for the damage and update its life.
- Signal SIGUSR1 must be used to exchange the fight mode between attack and defence.
- Signal SIGUSR2 must be used for looser to tell the winner that the fight has finished.

```
jordi@jordi-VirtualBox:~/pokemon$ ./ash
[13787] Pokedex is ready to search info...
[13787] Pokemons are ready to fight...
[13787] Fight starts...
[13790] ########Ninetales[13/13]
[13789] ########Kabutops[11/11]
[13789] ########Kabutops ---(Send attack with damage(1))---> Ninetales[13790]
[13790] ########Ninetales <---(Receives attack with damage(1))--- Kabutops[13789]
[13789] ########Kabutops[11/11]
[13790] ########Ninetales[12/13]
[13790] ########Ninetales ---(Send attack with damage(2))---> Kabutops[13789]
[13789] ########Kabutops <---(Receives attack with damage(2))--- Ninetales[13790]
[13790] ########Ninetales[12/13]
[13789] ########Kabutops[9/11]
[13789] ########Kabutops ---(Send attack with damage(3))---> Ninetales[13790]
[13790] ########Ninetales <---(Receives attack with damage(3))--- Kabutops[13789]
[13789] ########Kabutops[9/11]
[13790] ########Ninetales[9/13]
[13790] ########Ninetales ---(Send attack with damage(4))---> Kabutops[13789]
[13789] ########Kabutops <---(Receives attack with damage(4))--- Ninetales[13790]
[13790] ########Ninetales[9/13]
[13789] ########Kabutops[5/11]
[13789] ########Kabutops ---(Send attack with damage(1))---> Ninetales[13790]
[13790] ########Ninetales <---(Receives attack with damage(1))--- Kabutops[13789]
[13789] ########Kabutops[5/11]
[13790] ########Ninetales[8/13]
[13790] ########Ninetales ---(Send attack with damage(1))---> Kabutops[13789]
[13789] ########Kabutops <---(Receives attack with damage(1))--- Ninetales[13790]
[13790] ########Ninetales[8/13]
[13789] ########Kabutops[4/11]
[13789] ########Kabutops ---(Send attack with damage(1))---> Ninetales[13790]
[13790] ########Ninetales <---(Receives attack with damage(1))--- Kabutops[13789]
[13789] ########Kabutops[4/11]
[13790] ########Ninetales[7/13]
[13790] ########Ninetales ---(Send attack with damage(5))---> Kabutops[13789]
[13789] ########Kabutops <---(Receives attack with damage(5))--- Ninetales[13790]
Kabutops[13789] I have been defeated by Ninetales[13790]
[13789] ########Kabutops[0/11]
[13787] The fight ends...
jordi@jordi-VirtualBox:~/pokemon$
```

Notice: This design approach is an idea. Each group can be creative and modify the design to make the same purpose. It will be very constructive and positive to evaluate different designs to obtain similar results.

4 Pokemon daemon — (1 points)

> Do you know who I am? I'm Ash, from the town of Pallet. I'm destined to be the world's number one.

We will now use the code pokemond.c to create a daemon process that will simulate the context in which the games will be played.

- First of all, you need to understand what `pokemond.c` does in your machine. You will need to explain this in the final report.
- Modify the `Makefile` to compile and start the daemon.
- If the pokemon daemon is started then start ash.c. Otherwise, it must show an error: [ERROR] To play a pokemon game, the pokemon daemon must be started and now is stopped.

Notice: Observe and research the characteristics of a demon in Linux and the differences with a traditional process.

TIPS: Look at the kill manual, its operation and its return code.

Evaluation

1. I will focus on the correct use of system calls, errors handling and coding style.
2. The evaluation will be a face-to-face or virtual delivery (presentation) where the groups will present their solutions, and the class or the teacher can ask them questions about the code or the design.
3. Members of a group may have different grades depending on the degree of comprehension in the face-to-face assessment.