Jacob Little

ECE 561 (001)

Instructor: Dr. Michela Becchi

Project 1 Report

Introduction

This report discusses the implementation of three features to the Xinu operating system as well as answers to the questions presented in the project specification.

Question Answers

Q1. The maximum number of processes accepted by Xinu is 100 by default. This value is defined by `NPROC` in **config/Configuration**. The value defined in **config/Configuration** will overwrite the value defined in **config/conf.h** which will in turn it will be set to 8 by **include/process.h**.

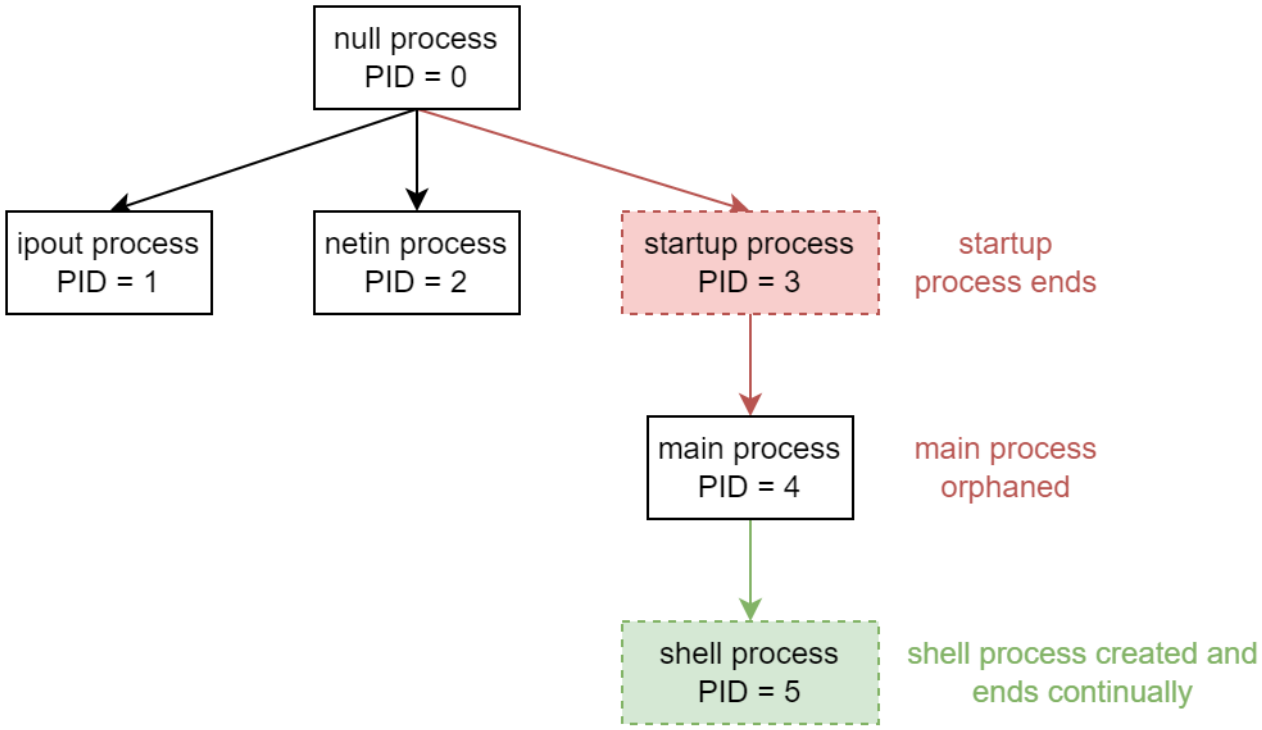
Q2. An inline definition in **include/process.h** called `isbadpid()` defines the criteria for an illegal (or "bad") PID. A PID is illegal if it is less than zero, greater than or equal to the maximum number of processes, or if it corresponds to a process table entry that is not currently being used (e.g. its state is `PR_FREE`).

Q3. The default stack size for a process is 65536 bytes which is defined by `INITSTK` in **include/process.h**.

Q4.

Q5. The shell process is created in **system/main.c** during the main process. It is created once first before the main process enters into a loop and continuously waits for the shell process to end so that it can recreate the shell process and start the cycle anew.

Q6. Process tree immediately after initialization:



Q7. `receive()` stalls the parent process until its child process ends execution.

P1. Timing

This problem involved modifying:

- **include/**
  - **process.h** - added time that process began execution to the process table entry data structure
  - **clock.h** - added `ctr1000` so that it could be used by other files
- **system/**
  - **clkhandler.c** - increment ctr1000 every 1 millisecond so that it can be used to track process time elapsed in milliseconds.
  - **create.c** - capture value of ctr1000 at process creation and store it in its process table entry
  - **initialize.c** - capture value of ctr1000 at null process creation and store it in its process table entry
- **shell/**
  - **xsh_ps.c** - added additional column to ps command printout that shows time since process began execution

In order to keep time as accurate as possible, the value of `ctr1000` is captured at the beginning of the `xsh_ps` function and then used later on to calculate the difference between process creation (which is stored in the process table), and the captured time from `ctr1000`.

P2. Process Creation and Stack Handling

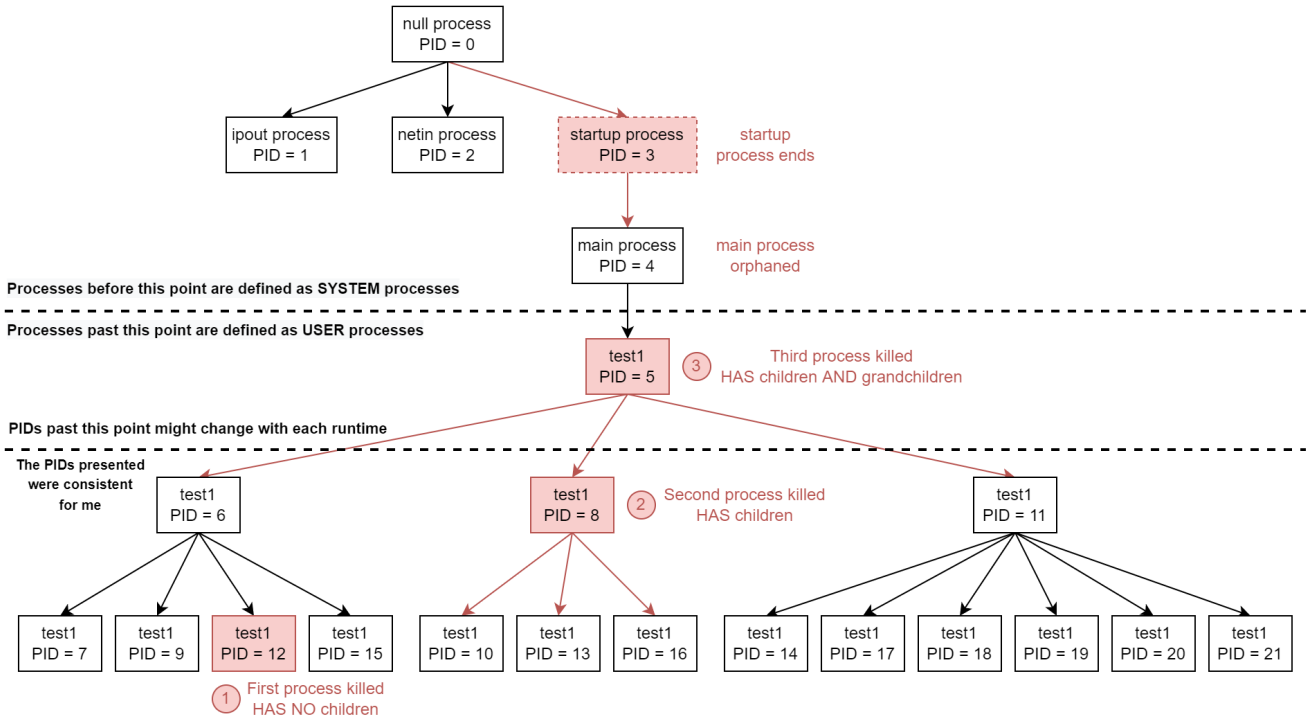
This problem involved modifying:

- **include/**
  - **prototypes.h** - added `syncprintf()` so that it could be used for debugging in **system/fork.c**. No instances of `syncprintf()` have been left in the final version of **system/fork.c**
- **system/**
  - **fork.c** - Implemented process fork

P3. Cascading Termination

This problem involved modifying:

- **include/**
  - **process.h** -
- **system/**
  - **create.c** -
  - **fork.c** -
  - **initialize.c** -
  - **kill.c** -
  - **main.term** -


null process
PID = 0
ipout process
PID = 1
netin process
PID = 2
startup process
PID = 3
startup
process ends
main process
PID = 4
main process
orphaned
Processes before this point are defined as SYSTEM processes
Processes past this point are defined as USER processes
test1
PID = 5
3
Third process killed
HAS children AND grandchildren
PIDs past this point might change with each runtime
The PIDs presented
were consistent
for me
test1
PID = 6
test1
PID = 8
2
Second process killed
HAS children
test1
PID = 11
test1
PID = 7
test1
PID = 9
test1
PID = 12
test1
PID = 15
test1
PID = 10
test1
PID = 13
test1
PID = 16
test1
PID = 14
test1
PID = 17
test1
PID = 18
test1
PID = 19
test1
PID = 20
test1
PID = 21
1
First process killed
HAS NO children