University of Cyprus
Department of Electrical and Computer Engineering

ECE 316 - Operating Systems and Networking Laboratory

Practical Assignment 6 (Due: 28/02/2019)

You should submit your report as well as the code for each of the exercises on blackboard. Your report must include the usual cover page. In your report, include any comments and description you may want to add. Naming format for the zip file: lastName_assignment6.zip. Caution: Remove the executables (.exe) from the files before you send them.

Παρατηρήσεις:

Σκοπός αυτής της άσκησης είναι η εξοικείωση με τις διεργασίες καθώς επίσης και με τις σωληνώσεις. Μπορούν να χρησιμοποιηθούν οι γλώσσες C/C++ για την εκπλήρωση των ασκήσεων.
Οι εντολές που θα χρησιμοποιηθούν είναι μοναδικές για λειτουργικά UNIX. Έτσι για την επιτυχή εκπλήρωση της άσκησης θα μπορούσατε να χρησιμοποιήσετε το λειτουργικό "Ubuntu" ή να ακολουθήσετε τα βήματα για την εγκατάσταση του εργαλείου Cygwin που προσφέρει παρόμοιες δυνατότητες με Unix/Linux λειτουργικά συστήματα σε περιβάλλον Window.
Μπορείτε να εγκαταστήσετε στο προσωπικό σας υπολογιστή λειτουργικό σύστημα UNIX ή LINUX. Σε περίπτωση που δεν είναι δυνατή η εγκατάσταση του η άσκηση μπορεί να υλοποιηθεί στους Η/Υ του εργαστηρίου του τμήματος (RED HAT). Σε αύτη την περίπτωση ενημερώστε τους βοηθούς του μαθήματος έτσι ώστε να διευθετηθεί ο τρόπος εξέτασης.
Ένα πιθανό λειτουργικό σύστημα που μπορείτε να εγκαταστήσετε είναι το UBUNTU που μπορείτε να το κατεβάσετε από: https://www.ubuntu.com/desktop/developers και να το εγκαταστήσετε παράλληλα με το λειτουργικό Windows ακολουθώντας τις εξής εντολές:
http://www.techsupportalert.com/content/easy-way-install-ubuntu-within-windows.htm ,
είτε https://www.lifewire.com/wubi-linux-installation-program-2201175

Παρακαλώ όπως διαβάσετε την τελευταία σελίδα αλλά και τα links που σας παρέχουμε τα οποία θα σας βοηθήσουν να υλοποιήσετε την άσκηση.

1. [25%]

Στην άσκηση αυτή καλείστε να δημιουργήσετε τρείς ξεχωριστές διεργασίες χρησιμοποιώντας την εντολή «fork()». Όλες οι διεργασίες πρέπει να δημιουργούνται από την κύρια/πατρική διεργασία. Η κάθε διεργασία πρέπει να εκτυπώνει στην οθόνη 15 τουλάχιστον μηνύματα τα οποία περιλαμβάνουν τον αριθμό της και τον αριθμό του μηνύματος (e.g Process 1 message 1)

Ερωτήσεις:

- Τι παρατηρείτε στην σειρά εκτύπωσης των μηνυμάτων?
- Σε τι αντιστοιχεί η τιμή που επιστρέφει η επιτυχής ολοκλήρωση της εντολής «fork()»?
- Μετά από δύο συνεχόμενες κλήσεις της «fork()» σε ένα πρόγραμμα, πόσες συνολικά διεργασίες υπάρχουν ?
- Η αλλαγή στις τιμές των μεταβλητών μιας διεργασίας επηρεάζει τις τιμές των μεταβλητών μιας άλλης διεργασίας του ίδιου προγράμματος?

2. [25%]

   Στην άσκηση αυτή επαναλάβετε την ‘Άσκηση 1 αλλά αυτήν την φορά χρησιμοποιήστε την εντολή “sleep()” για να εισάγετε καθυστέρηση στην εκτύπωση των μηνυμάτων. Η καθυστέρηση πρέπει να υπολογίζεται σε σχέση με τον αριθμό της διεργασίας αλλά και τον αριθμό του μηνύματος που εκτυπώνεται την συγκεκριμένη στιγμή. Μπορείτε να χρησιμοποιήσετε όποια αναλογία θέλετε φτάνει να είναι η ίδια για όλες τις διεργασίες.

3. [25%]

   Δημιουργήστε ένα πρόγραμμα με 3 διεργασίες που η κάθε μία εκτυπώνει συνολικά 15 μηνύματα. Χρησιμοποιώντας τις σωληνώσεις (μία σωλήνωση για κάθε διεργασία) καθορίστε την σειρά εκτύπωσης των διεργασιών οι οποίες να τυπώνουν μόνο ένα μήνυμα σε κάθε κύκλο. Για παράδειγμα αν επιλέξουμε να εκτυπώνει ένα μήνυμα η διεργασία 1, ένα η διεργασία 3 και ένα η διεργασία 2 σε κάθε κύκλο, τότε το αποτέλεσμα θα είναι το ακόλουθο:

   Process 1 message 1
   Process 3 message 1
   Process 2 message 1
   Process 1 message 2
   Process 3 message 2
   Process 2 message 2
   …

   Βοήθεια:

   Η πατρική διεργασία θα πρέπει να γράφει στην σωλήνωση από την οποία διαβάζει η διεργασία 1. Η διεργασία 1 θα διαβάζει την σωλήνωση αυτήν και θα γράφει στην επόμενη κ.ο.κ. Επίσης σε κάθε ανάγνωση και γραφή όλες οι σωληνώσεις που δεν χρησιμοποιούνται πρέπει να είναι κλειστές.

4. [25%]

   Στην άσκηση αυτή καλείστε να αλλάξετε τον κώδικα της άσκησης 3 έτσι ώστε να μπορεί να δεχτεί N διεργασίες. Ο ακριβής αριθμός των διεργασιών, N, να καθορίζεται από το πρόγραμμα μέσω μιας σταθεράς.

   Ερωτήσεις:

   - Πως καταλαβαίνει μία διεργασία το πότε πρέπει να διαβάσει από μία σωλήνωση?
   - Πιο πρόβλημα προκύπτει όταν η πατρική διεργασία τελειώσει πριν τις υπόλοιπες διεργασίες του προγράμματος?

Processes – Contex Information:

- Process ID (pid) – *unique integer*
- Parent Process ID (ppid)
- Real User ID – ID of user/process started
- Effective User ID – ID of user who wrote the process program
- A process ID or *pid (getpid())* is a positive integer that uniquely identifies a running process, and is stored in a variable of type *pid_t(getppid())*.

1. **fork():**
   - Creates a child process by making a copy of the parent process --- an exact duplicate.
   - Since a child process is a copy of the parent, it has copies of the parent's data.
   - A change to a variable in the child will *not* change that variable in the parent.
   - Both the child *and* the parent continue running.

   I. **pid = fork();**
   - In the child: **pid == 0;** In the parent: **pid == the process ID of the child.**
   - A program almost always uses this pid difference to do different things in the parent and child.

   II. **wait():**
   - Suspends the calling process until one of its child processes ends.
   - A process that calls wait() can:
     - ***suspend* (block)** if all of its children are still running, or
     - ***return* immediately** with the ***termination* status** of **a** child, or
     - ***return* immediately** with an ***error*** if there are no child processes.

   III. **pipe():**
   - A Unix pipe provides a one-way flow of data.
   - Pipes allow you to chain together commands in a very elegant way, passing output from one program to the input of another to get a desired end result.
   - A pipe can be explicitly created in Unix using the *pipe* system call
   - Two file descriptors are returned--fildes[0] and fildes[1], and they are both open for reading and writing.
   - A read from fildes[0] accesses the data written to fildes[1] on a first-in-first-out (FIFO) basis and a read from fildes[1] accesses the data written to fildes[0] also on a FIFO basis.

Πιο κάτω μπορείτε να βρείτε μερικά links τα οποία σας παρέχουν μερικά παραδείγματα τα οποία μπορείτε να χρησιμοποιήσετε για να υλοποιήσετε τις ασκήσεις:

http://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/create.html

https://www.geeksforgeeks.org/fork-system-call/

http://tldp.org/LDP/lpg/node11.html

https://www.geeksforgeeks.org/pipe-system-call/