

K24: Προγραμματισμός Συστήματος  
2η Εργασία – Εαρινό Εξάμηνο 2017  
Προθεσμία Υποβολής: Κυριακή 30 Απριλίου 2017 Ώρα 23:59

### Εισαγωγή στην Εργασία:

Ο στόχος της εργασίας αυτής είναι να εξοικειωθείτε με την δημιουργία διεργασιών (processes) με χρήση των κλήσεων συστήματος fork/exec, την επικοινωνία διεργασιών μέσω named-pipes/pipes, τη χρήση low level I/O, το χειρισμό signals, και τη δημιουργία bash scripts.

Καλείστε να υλοποιήσετε ένα πρόγραμμα που υλοποιεί τη λειτουργία ενός συστήματος διαχείρισης εργασιών, job management system (jms), στο οποίο θα υποβάλλονται εργασίες χρήστη. Την εκτέλεση των εργασιών αυτών αναλαμβάνει μια ιεραρχία διεργασιών που δημιουργούνται δυναμικά με την βοήθεια των κλήσεων συστήματος fork() και exec(). Το jms θα μπορεί να διαχειρίζεται και να εκτελεί ταυτόχρονα ένα οποιαδήποτε αριθμό από υποβληθείσες εργασίες.

Η υλοποίησή σας θα πρέπει να αποτελείται από δύο βασικά προγράμματα, το jms\_coord και το jms\_console. Το πρώτο πρόγραμμα που είναι και ο συντονιστής εκτελείται στην αρχή και παραμένει συνεχώς σε λειτουργία, ενώ το δεύτερο εκτελείται από τον χρήστη για υποβολή εργασιών, τον έλεγχο τους, κλπ. Η επικοινωνία μεταξύ αυτών των δύο βασικών διεργασιών επιτυγχάνεται με τη χρήση named-pipes.

Τέλος θα πρέπει να δημιουργήσετε ένα bash script για την εξαγωγή στατιστικών στοιχείων για τις εργασίες που εκτελέστηκαν/εκτελούνται.

### Συνοπτική Περιγραφή:

Το Σχήμα 1 παρουσιάζει την οργάνωση και την αλληλεπίδραση εμπλεκόμενων διεργασιών στην υλοποίηση του jms όπως επίσης και τη συλλογή αποτελεσμάτων με την χρήση του script.

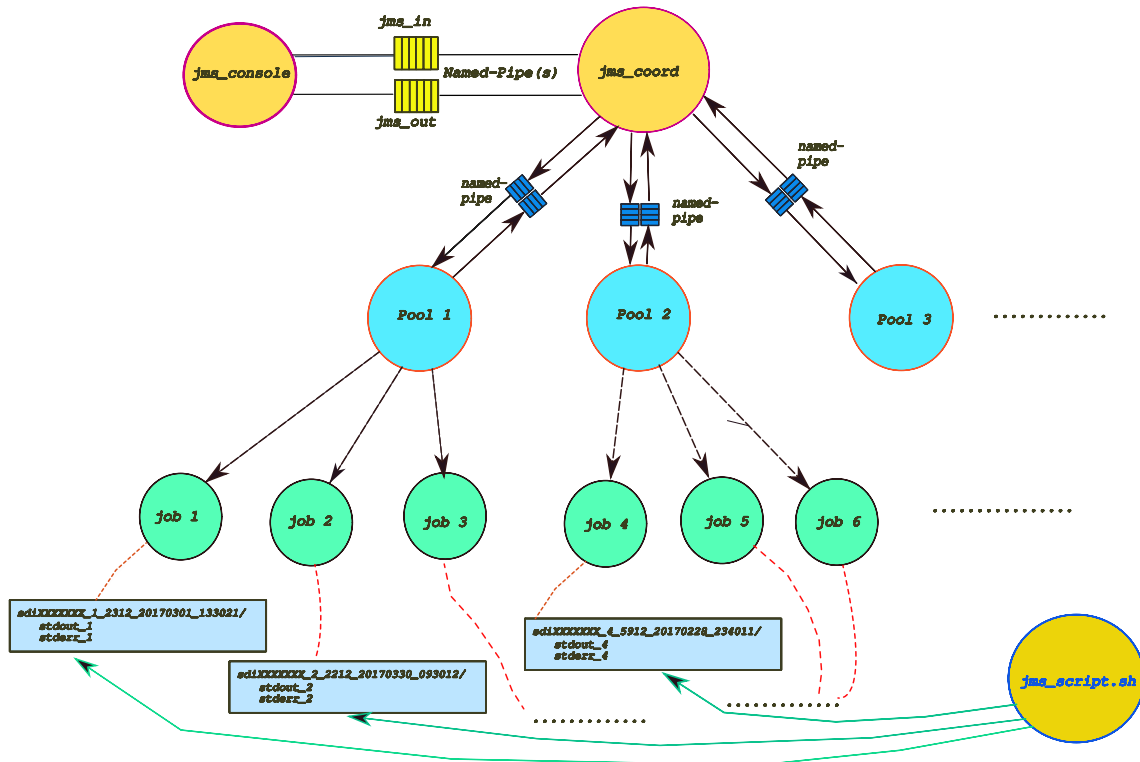
Ο συντονιστής jms\_coord διαχειρίζεται τις εργασίες χρήστη μέσω μια ιεραρχίας που δημιουργεί και επίσης παρακολουθεί την εξέλιξη αυτών των διεργασιών (δηλ. αν είναι ακόμη ενεργές ή αν έχουν τερματίσει). Η jms\_coord δεν «διεκπεραιώνει κατευθείαν» τις jobs που έχει καταθέσει ο χρήστης. Αντιθέτως, δυναμικά δημιουργεί κόμβους «δεξαμενές» (pools) που η κάθε μία μπορεί να διαχειρίζεται ένα συγκεκριμένο μέγιστο αριθμό από jobs χρηστών. Οι δεξαμενές δημιουργούνται δυναμικά και έτσι μπορούν να εξυπηρετήσουν συνολικά οποιοδήποτε αριθμό από jobs που έχουν κατατεθεί στον συντονιστή. Όταν μία δεξαμενή ολοκληρώσει τον μέγιστο αριθμό jobs που έχει αναλάβει απλά «παραδίδει» οποιαδήποτε στοιχεία έχει συλλέξει στη διεργασία jms\_coord και στην συνέχεια ολοκληρώνει την εκτέλεσή της αποχωρώντας από την ιεραρχία.

Η επικοινωνία του jms\_console με το jms\_coord πραγματοποιείται μέσω ζεύγους named-pipes τα ονόματα των οποίων μπορεί να είναι jms\_in, jms\_out. Η επικοινωνία μεταξύ pools και του jms\_coord πραγματοποιείται επίσης με named pipes με ονόματα που θα σχετίζονται με την άσκηση. Και οι δύο αυτές επικοινωνίες χρειάζονται κάποια (απλά) πρωτόκολλα ανταλλαγής πληροφορίας που θα πρέπει να τα σχεδιάσετε ώστε να εξυπηρετήσουν τις ανάγκες της εργασίας σας.

Το πρόγραμμα jms\_console είναι η διεπαφή του χρήστη με την υπηρεσία υποβολής εργασιών (δηλ. με τον jms\_coord). Το jms\_console αναλαμβάνει:

- την επικοινωνία με το jms\_coord μέσω των named-pipes.
- την υποβολή των εργασιών.
- την υποβολή ερωτήσεων σχετικά με την κατάσταση (status) της υπηρεσίας (δηλ. πόσες εργασίες εκτελούνται).

- την παρουσίαση των πληροφοριών που παράγει ο `jms_coord` για την/τον χρήστη.



Σχήμα 1: Οργανωτική διάταξη και αλληλεπίδραση διεργασιών για την υλοποίηση του `jms`

### Κλήση της `jms_console`:

Το πρόγραμμα εκτελείται από τη γραμμή εντολών ως εξής:

```
$ ./jms_console -w <jms_in> -r <jms_out> -o <operations_file>
```

όπου:

1. `jms_in` είναι το όνομα του `named-pipe` για εισαγωγή (εγγραφή) δεδομένων στον `jms_coord`.
2. `jms_out` είναι το όνομα του `named pipe` για έξοδο (ανάγνωση) δεδομένων προερχόμενα από τον `jms_coord`.
3. `operations_file` είναι το όνομα του αρχείου που περιέχει εντολές που πρέπει να σταλούν και να διεκπεραιωθούν από τη διεργασία `jms_coord` και την κάτω από αυτή ιεραρχία διεργασιών.

Σε περίπτωση που ολοκληρωθεί η εκτέλεση των εντολών που βρίσκονται στο `operations_file` ή το συγκεκριμένο όρισμα δεν υπάρχει στη γραμμή εντολών, το πρόγραμμα `jms_console` διαβάζει εντολές από τη γραμμή εντολών (standard input).

### Κλήση του `jms_coord`:

Το πρόγραμμα εκτελείται από τη γραμμή εντολών ως εξής:

```
$ ./jms_coord -l <path> -n <jobs_pool> -w <jms_out> -r <jms_in>
```

όπου

1. `path` είναι το όνομα του καταλόγου κάτω από τον οποίο θα δημιουργούνται όλα αρχεία ή κατάλογοι παράγονται στην διάρκεια εκτέλεσης των προγραμμάτων σας.
2. `jobs_pool` είναι ο μέγιστος αριθμός εργασιών που μπορεί να εκτελείται σε κάθε `pool`.

3. `jms_in` και `jms_out` είναι τα ονόματα των `named-pipes` που χρησιμοποιούνται για επικοινωνία με την `jms_console`.

### Περιγραφή του `jms_script.sh`:

Το πρόγραμμα εκτελείται από τη γραμμή εντολών ως εξής:

```
$ ./jms_script.sh -l <path> -c <command>
```

όπου

1. `path` είναι το όνομα του καταλόγου κάτω από τον οποίο έχουν δημιουργηθεί όσα αρχεία ή κατάλογοι απαιτούνται (βλέπε παραπάνω).
2. `command` είναι μία εντολή από τις εξής:
  - `list`: παρουσιάζει μία λίστα από τους καταλόγους που δημιούργησαν οι εργασίες
  - `size [n]`: παρουσιάζει τη λίστα από τους καταλόγους που δημιούργησαν οι εργασίες ταξινομημένη κατά αύξουσα σειρά μεγέθους των αρχείων που περιέχουν. Προαιρετικά μπορεί να παίρνει όρισμα ένα ακέραιο αριθμό `n`, που να ορίζει την εμφάνιση μόνο των `n` μεγαλύτερων τιμών.
  - `purge`: διαγράφει όλους τους υφιστάμενους καταλόγους που δημιούργησαν οι εργασίες.

Η σειρά με την οποία μπορούν να καλούνται οι σημαίες και οι αντίστοιχοι παράμετροι σε όλες τις παραπάνω διεπαφές, δεν είναι προκαθορισμένη.

### Διεπαφή με τον/την Χρήστη

Όταν τρέξει το `./jms_coord`, δέχεται εντολές από το `named-pipe jms_in` και επιστρέφει αποτελέσματα στο `named pipe jms_out` ως εξής:

#### 1. `submit <job>`

Η εντολή αυτή δέχεται την εργασία `job` προς εκτέλεση. Η διεργασία `jms_coord` βρίσκει το επόμενο διαθέσιμο `pool` (και το δημιουργεί αν δεν υπάρχει ήδη) και στέλνει την `job` για εκτέλεση. Η διεργασία `jms_coord` επιστρέφει τον λογικό αριθμό της εργασίας και το `pid` που έλαβε. Για παράδειγμα: `JobID: 3, PID: 2324`.

#### 2. `status <JobID>`

Η εντολή αυτή αναζητάει την κατάσταση της εργασίας με λογικό αριθμό `JobID`. Επιστρέφεται η κατάσταση της εργασίας. Παραδείγματα για τρεις πιθανές διαφορετικές εξόδους:

```
JobID 3 Status: Active (running for 3 seconds)
```

```
JobID 3 Status: Finished
```

```
JobID 3 Status: Suspended
```

#### 3. `status-all [time-duration]`

Η εντολή αυτή αναζητάει την κατάσταση όλων των εργασιών. Η εντολή μπορεί να έχει προαιρετικό όρισμα `time-duration` που να ορίζει μέχρι πόσα δευτερόλεπτα πριν θα πρέπει να έχουν υποβληθεί οι εργασίες.

Επιστρέφονται οι καταστάσεις των εργασιών. Παράδειγμα:

```
1. JobID 2 Status: Finished
```

```
2. JobID 3 Status: Active (running for 3 sec)
```

```
3. JobID 4 Status: Suspended
```

#### 4. `show-active`

Η εντολή αυτή αναζητάει τις εργασίες που είναι ενεργές. Επιστρέφονται οι λογικοί αριθμοί των εργασιών και η ένδειξη `Active`. Παράδειγμα:

```
Active jobs:
```

1. JobID 3
2. JobID 11

#### 5. show-pools

Η εντολή αυτή αναζητάει τις διεργασίες-δεξαμενές με τα Linux pids τους όπως επίσης και των αριθμό των jobs που η κάθε μία έχει 'υπό εκτέλεση' αυτή την στιγμή. Επιστρέφονται οι αριθμοί των εργασιών που υπάρχουν σε κάθε δεξαμενή. Παράδειγμα:

Pool & NumOfJobs:

1. 2345 3
2. 2356 11
3. 3456 2

#### 6. show-finished

Η εντολή αυτή αναζητάει τις εργασίες που έχουν ολοκληρωθεί. Επιστρέφονται οι αριθμοί των εργασιών και ένδειξη Finished. Παράδειγμα:

Finished jobs:

1. JobID 4
2. JobID 6

#### 7. suspend <JobID>

Η εντολή αυτή αναστέλλει την εκτέλεση της εργασίας (λογικό) <JobID>. Επιστρέφεται η ανακοίνωση ότι εστάλη σήμα στη συγκεκριμένη εργασία. Παράδειγμα:

Sent suspend signal to JobID 3

#### 8. resume <JobID>

Η εντολή αυτή επαναφέρει την εκτέλεση της εργασίας <JobID>. Επιστρέφεται ένα μήνυμά ότι εστάλη σήμα στη συγκεκριμένη εργασία. Παράδειγμα:

Sent resume signal to JobID 3

#### 9. shutdown

Η εντολή αυτή τερματίζει την εκτέλεση της υπηρεσίας με την εξής σειρά: Η διεργασία jms\_coord στέλνει σήματα SIGTERM στα pools, τα οποία το αναχαιτίζουν με κατάλληλο signal handler, στέλνουν μήνυμα τερματισμού (SIGTERM) στις εργασίες που επιτηρούν, αναφέρουν ό,τι απαιτείται στη διεργασία jms\_coord και τερματίζουν. Η διεργασία jms\_coord μετά των τερματισμό όλων των pools τερματίζει με τη σειρά της επιστρέφοντας συνοπτικά στατιστικά στοιχεία. Παράδειγμα:

Served 19 jobs, 3 were still in progress

Κάθε job θα δημιουργεί ένα κατάλογο (directory), στον οποίο θα αποθηκεύει τα αποτελέσματα της εκτέλεσης του. Το όνομα του καταλόγου θα είναι sdixxxxxx\_jobid\_pid\_date\_time, όπου:

- sdixxxxxx το account name σας
- jobid το λογικό αναγνωριστικό της εργασίας που εκτελείται
- pid το Linux pid της εργασίας που εκτελείται
- date η ημερομηνία σε μορφή 20170331 για 31 Μαρτίου 2017.
- time η ώρα σε μορφή 175000 για 17:50:00.

Ένα παράδειγμα τέτοιου καταλόγου είναι το: jms\_sdi0100001\_3\_20170331\_142123. Ο κατάλογός θα δημιουργείται με την έναρξη εκτέλεσης της εργασίας και θα περιέχει τουλάχιστον τα εξής αρχεία:

1. stdout\_jobid, όπου θα περιέχεται η έξοδος της εργασίας στο standard output.
2. stderr\_jobid, όπου θα περιέχεται η έξοδος της εργασίας στο standard error.

### Διαδικαστικά:

- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C (ή C++ αλλά χωρίς την χρήση έτοιμων δομών της standard βιβλιοθήκης/STL) και να τρέχει στα Linux workstations του τμήματος.
- Το πρόγραμμα σας (source code) πρέπει να αποτελείται από **τουλάχιστον** δυο (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Η χρήση του separate compilation είναι **επιτακτική!**
- Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα περιλαμβάνονται στα κριτήρια βαθμολόγησης.
- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι <https://piazza.com/uo.gr/spring2017/k24/home>. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.

### Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1-2 σελίδες ASCII κειμένου είναι αρκετές) σε αρχείο README.
2. Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο ή αλλού θα πρέπει να αναφερθεί και στον πηγαίο κώδικά σας αλλά και στο παραπάνω README.
3. Όλη η δουλειά σας σε ένα tar-file (ΌνομαEponymoProject2.tar) που να περιέχει όλα τα source files, header files, Makefile.

### Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι ατομικές.
2. Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο μηδενίζεται στο μάθημα.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμειγμένος σε αντιγραφή κώδικά απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.