Κέλυφος Bash – Τι μάθαμε?

Μεταξύ άλλων…

- Συνθήκες ελέγχου (if, case…) και βρόχοι επανάληψης (for, while)
- Πράξεις ακεραίων (let, expr) και δεκαδικών (bc)
- Επεξεργασία string
  - expr substr string position length
  - expr length string
- Διάβασμα λέξεων με read, set, πίνακες
- Και πολλά άλλα…

Θέμα 1

Γράψτε ένα πρόγραμμα για το κέλυφος Bourne (έστω ότι ονομάζεται "encol") το οποίο να διαβάζει την είσοδό του από την προκαθορισμένη είσοδο, να γράφει την έξοδό του στην προκαθορισμένη έξοδο και το οποίο, με βάση αριθμούς στηλών που του δόθηκαν από τη γραμμή εντολής, να κατασκευάζει την έξοδό του σαν παράθεση των χολονών από την είσοδο που ορίζονται από τους δεδομένους αριθμούς στηλών. Δηλαδή, αν το "encol" κληθεί σαν "encol $\langle col.1 \rangle$ $\langle col.2 \rangle$ ... $\langle col.N \rangle$", στην έξοδο να εκτυπωθούν πρώτα οι στήλες 1 έως $\langle col.1 \rangle - 1$ της εισόδου, μετά οι στήλες $\langle col.1 \rangle$ έως $\langle col.2 \rangle - 1$ και ούτω καθεξής και τέλος οι στήλες $\langle col.N \rangle$ έως την τελευταία. Φυσικά, αν κάποια γραμμή της εισόδου δεν είναι αρκετά πλατιά, οι μη υπάρχουσες χολόνες αυτής της γραμμής θα πρέπει να προκαλούν την εγγραφή στην έξοδο κενών γραμμών. Με άλλα λόγια, ανεξάρτητα από το πλάτος των γραμμών στην είσοδο, αν αυτές είναι $\langle M \rangle$ στο πλήθος, η έξοδος πρέπει να έχει $\langle M \rangle \times (\langle N \rangle + 1)$ γραμμές. Ένα αρχείο εισόδου και ένα παράδειγμα εκτέλεσης του "encol" φαίνονται στα σχήματα:

```
$ cat inp_file
1. 234 George B. 4. 334 Helen B.
2. 247 Julia R.
3. 327 Robert R. 6. 389 Mary C. 9. 480 Peter G.
$
```

```
$ ./encol 18 33 < inp_file
1. 234 George B.
2. 247 Julia R.
3. 327 Robert R.
4. 334 Helen B.

6. 389 Mary C.



9. 480 Peter G.
$
```

Λύση – Θέμα 1

```
#!/bin/bash

start=1
end=0
last=10000000
#save at a temp file
touch myfile
while read line
do
echo "$line" >> myfile
done

for str
do
first=$str
if [ $end -gt 0 ]
then
echo "`cat myfile | colrm $first $last | colrm $start $end`"
else
echo "`cat myfile | colrm $first $last`"
fi

let "end=$str-1"
done
#We now need to print all columns after last given number
echo "`cat myfile | colrm $start $end`"

rm -f myfile
```

Θέμα 2

Γράψτε ένα πρόγραμμα για το κέλυφος Bourne (έστω ότι ονομάζεται "reformat") το οποίο, εκτελούμενο με μία παράμετρο $\langle n \rangle$ στη γραμμή εντολής, να διαβάζει από την προκαθορισμένη είσοδο (stdin) ένα κείμενο δομημένο σε γραμμές από λέξεις (οι λέξεις χωρίζονται με έναν η περισσότερους κενούς χαρακτήρες) και να το ξαναγράφει στην προκαθορισμένη έξοδο (stdout) έτσι ώστε καμία γραμμή να μην ξεπερνά τους $\langle n \rangle$ χαρακτήρες, μεταξύ των λέξεων να υπάρχει ακριβώς ένας κενός χαρακτήρας, όλες οι γραμμές να αρχίζουν με μη κενό χαρακτήρα και να μην τελειώνουν με κενούς χαρακτήρες. Υποθέστε ότι στην είσοδό σας δεν υπάρχει λέξη με περισσότερους από $\langle n \rangle$ χαρακτήρες. Ένα παράδειγμα εκτέλεσης είναι το ακόλουθο:

```
$ cat file.txt
   This is a text file to
test   the Bourne script for
reformatting.  Use it    with
  care!
$ ./reformat 17 < file.txt
This is a text
file to test the
Bourne script for
reformatting. Use
it with care!
$
```

Λύση – Θέμα 2

```
#!/bin/bash

limit=$1
left=$limit
startLine=1
while read line
do
  set -- $line
  for str
  do
    x=`expr length $str`
    if [ $startLine -eq 0 ]
    then
      let "x=x+1"
    fi

    if [ "$x" -gt "$left" ]
    then
      echo
      startLine=1
      let "left=limit"
    fi
    if [ "$startLine" -eq 0 ]
    then
      echo -n " $str"
      let "left=left-x"
    else
      echo -n "$str"
      x=`expr length $str`
      let "left=left-x"
      startLine=0
    fi
done
done
echo
```

Επικοινωνία Διεργασιών

Έστω ότι θέλετε να υλοποιήσετε μία εφαρμογή στην οποία μία διεργασία *P1* πρέπει να στέλνει δεδομένα σε μία διεργασία *P2* και μία διεργασία *P3* να στέλνει δεδομένα σε μία διεργασία *P4*. Κάτω από ποιες προϋποθέσεις θα μπορούσε η προηγούμενη επικοινωνία να επιτευχθεί μέσω σωλήνων; Θα αρκούσε ένας σωλήνας ή χρειάζονται περισσότεροι; Θα μπορούσαν να χρησιμοποιηθούν υποδοχές ροής για να επιτευχθεί αυτή η επικοινωνία και, αν ναι, σε ποιο πεδίο; Θα μπορούσατε να χρησιμοποιήσετε ουρές μηνυμάτων; Αν ναι, αρκεί μία ή χρειάζονται περισσότερες; Πώς θα σχολιάζατε τη χρήση κοινής μνήμης για την επικοινωνία αυτή;

Νήματα Εργάτες

Σε μία πολυνηματική εφαρμογή, το αρχικό νήμα τοποθετεί στοιχεία προς επεξεργασία σε μία καθολικά προσπελάσιμη ουρά και, στη συνέχεια, δημιουργεί NWorkers νήματα-εργάτες, τα οποία διαχειρίζονται την ουρά για να επεξεργασθούν τα δεδομένα που περιέχει. Κάθε εργάτης, όταν δεν έχει δουλειά, εξάγει κάποιο στοιχείο από την ουρά, το επεξεργάζεται και, αναλόγως με το αποτέλεσμα της επεξεργασίας, ενδέχεται να προσθέσει στην ουρά και κάποιο νέο στοιχείο για επεξεργασία. Μετά, συνεχίζει παίρνοντας νέο στοιχείο από την ουρά για επεξεργασία. Η κοινή προσπάθεια των εργατών τερματίζει όταν δεν υπάρχει, και είναι βέβαιο ότι δεν θα υπάρξει, άλλο στοιχείο στην ουρά για επεξεργασία. Το απόσπασμα κώδικα C που φαίνεται στη συνέχεια προέρχεται από τη συνάρτηση των νημάτων-εργατών. Συμπληρώστε το απόσπασμα αυτό, όπου υπάρχουν αποσιωπητικά, για να επιτευχθεί η επιθυμητή συμπεριφορά. Μπορείτε να χρησιμοποιήσετε και επιπλέον, εξωτερικές ή όχι, μεταβλητές, όχι όμως σηματοφόρους ή μεταβλητές συνθήκης, πλην αυτών που περιλαμβάνονται ήδη στο απόσπασμα. Ό,τι συναρτήσεις χρησιμοποιούνται, θεωρήστε ότι έχουν οριστεί κατάλληλα.

```
while (!pthread_mutex_lock(&mtx)) {
   if (is_empty_queue(queue)) {
.....................................
      pthread_cond_wait(&cvar, &mtx);
.....................................
      continue; }
   item = remove_from_queue(queue);
.....................................
   newitem = process_item(item);
   if (needs_processing(newitem)) {
.....................................
      insert_in_queue(queue, newitem);
..................................... } }
```

Όρια Μηνυμάτων

Γνωρίζετε ότι στην επικοινωνία μεταξύ διεργασιών μέσω σωλήνων ή υποδοχών ροής δεν υποστηρίζονται από το ίδιο το πρωτόκολλο όρια μεταξύ των μεταδιδόμενων μηνυμάτων. Προτείνετε επιγραμματικά (σε **6 γραμμές το πολύ**) τρεις τρόπους για να υλοποιήσει κανείς όρια μεταξύ μηνυμάτων, σε μία εφαρμογή που χρησιμοποιεί αυτούς τους τρόπους επικοινωνίας. Γράψτε, για έναν από αυτούς τους τρόπους, και μόνο για την παραλαβή μηνυμάτων, τμήμα προγράμματος C που εγγυάται την ανάγνωση ενός αυτοτελούς μηνύματος.

Τι κάνει το παρακάτω?

Μελετήστε το τμήμα προγράμματος C στο διπλανό σχήμα και εξηγήστε πολύ συνοπτικά τη λειτουργία του. Αν το εκτελέσιμο πρόγραμμα που προκύπτει απ' αυτό ονομάζεται "hide", συμπληρώστε τα αποτελέσματα των εντολών που φαίνονται στη συνέχεια. Υποθέστε ότι όλες οι κλήσεις συστήματος θα είναι επιτυχείς.

```
$ wc -c hide.c
616 hide.c
$ cat inp.txt
abcdefghijklmno
ABCDEFGHIJKLMNO
123456789012345
678901234567890
$ ./hide 1 < inp.txt
.....
$ ./hide 3 < inp.txt
.....
$ ./hide 8 < hide.c | wc -c
.....
```

```
.....
main(int argc, char *argv[])
{  int i, n, status, fd[2];
   int flag = 1; char ch;
   n = atoi(argv[1]);
   for (i=0 ; i<n ; i++) {
      if (i != n-1) pipe(fd);
      if (fork() == 0) {
         if (i != n-1) {
            close(fd[0]);
            dup2(fd[1], 1);
            close(fd[1]); }
         while (read(0, &ch, 1) > 0) {
            if (flag < 0) write(1, &ch, 1);
            flag = -flag; }
         exit(0); }
      else {
         if (i != n-1) {
            dup2(fd[0], 0);
            close(fd[0]);
            close(fd[1]); } } }
.....
```

Επιστροφή Τιμών

"Όταν τερματίσει μία διεργασία, έχει τη δυνατότητα να επιστρέψει έναν ακέραιο κωδικό εξόδου στη διεργασία που την δημιούργησε, και μόνο σ' αυτήν. Το ίδιο ακριβώς ισχύει και για τα νήματα, δηλαδή όταν τερματίζει ένα νήμα, μπορεί να επιστρέψει έναν ακέραιο κωδικό εξόδου στο νήμα που το δημιούργησε, και μόνο σ' αυτό. Η μόνη διαφορά είναι ότι πρέπει να επιστραφεί ο ακέραιος αυτός κωδικός προσαρμοσμένος σε δείκτη σε `void`*. Επίσης, όταν δημιουργείται ένα νήμα, υπάρχει η δυνατότητα να περάσουμε στη συνάρτηση που θα αρχίσει να εκτελεί το νήμα μία παράμετρο. Αυτό όμως είναι αρκετά περιοριστικό, γιατί κάποιες φορές θα θέλαμε να περάσουμε στη συνάρτηση περισσότερες παραμέτρους, οπότε σ' αυτές τις περιπτώσεις η μόνη μας λύση είναι η χρήση εξωτερικών μεταβλητών."* Σχολιάστε τα προηγούμενα σε **20 γραμμές το πολύ.**

Τι κάνει το παρακάτω?

Αν το εκτελέσιμο πρόγραμμα που προκύπτει από το πρόγραμμα C στην επόμενη σελίδα ονομάζεται "ptt", δώστε ένα πιθανό αποτέλεσμα της εντολής "ptt 2 3". Εξηγήστε επίσης, πολύ συνοπτικά, ποια είναι η λειτουργία του προγράμματος.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int m;
void *f(void *argp)
{  pthread_t *thrs;
   int i, retcode, sum = 0, n = (int) argp;
   thrs = malloc(m * sizeof(pthread_t));
   printf("%d %d\n", n, pthread_self());
   if (n--) {
      for (i=0 ; i<m ; i++)
         pthread_create(thrs+i, NULL, f, (void *) n);
      for (i=0 ; i<m ; i++) {
         pthread_join(*(thrs+i), (void **) &retcode);
         sum += retcode; }
      pthread_exit((void *) sum); }
   pthread_exit((void *) pthread_self()); }

main(int argc, char *argv[])
{  pthread_t thr;
   int n, retcode;
   n = atoi(argv[1]);  m = atoi(argv[2]);
   pthread_create(&thr, NULL, f, (void *) n);
   pthread_join(thr, (void **) &retcode);
   printf("%d\n", retcode);
   pthread_exit(NULL); }
```

Ποιο θα είναι το αποτέλεσμα της εκτέλεσης της εντολής "ptt 3 5 | wc -l"; Γενικά, τι θα περιμένατε να εκτυπώνεται από την εντολή "ptt ⟨n⟩ ⟨m⟩ | wc -l", για οποιαδήποτε ⟨n⟩ και ⟨m⟩, και γιατί;

Επικοινωνία Διεργασιών/Νημάτων

"Όταν μία γονική διεργασία δημιουργεί διεργασίες-παιδιά, δεν είναι δυνατόν να επιτύχουμε επικοινωνία μεταξύ γονέα και παιδιών μέσω εξωτερικών/καθολικών μεταβλητών, γιατί οι μεταβλητές αυτές αντιστοιχίζονται σε διαφορετικές θέσεις μνήμης κατά τη δημιουργία των παιδιών. Είναι δυνατόν όμως να πετύχουμε κάτι τέτοιο αν κάνουμε δυναμική δέσμευση μνήμης στο γονέα (με `malloc`*), γιατί ο χώρος στον οποίο δεσμεύεται αυτή (σωρός) είναι κοινός για το γονέα και για τα παιδιά. Αντίστροφα, όταν σε μία διεργασία το αρχικό νήμα δημιουργήσει ένα πλήθος από άλλα νήματα, κάθε νήμα έχει διαφορετικό σωρό, οπότε δεν μπορούμε να πετύχουμε επικοινωνία μεταξύ των νημάτων με δυναμική δέσμευση μνήμης. Ενώ, με εξωτερικές/καθολικές μεταβλητές, μπορούμε να έχουμε επικοινωνία μεταξύ των νημάτων, αφού ο χώρος που φυλάσσονται αυτές είναι κοινός για όλα τα νήματα μίας διεργασίας."* Σχολιάστε τα προηγούμενα σε **5 γραμμές το πολύ.**

Σηματοφόροι/Κοινή Μνήμη

Δίνεται στη συνέχεια τμήμα C προγράμματος, το εκτελέσιμο του οποίου ονομάζεται "prshmem". Δώστε ένα πιθανό αποτέλεσμα της εντολής "prshmem 5". Αιτιολογήστε συνοπτικά την απάντησή σας.

```
main(int argc, char *argv[]) { ..............................
  struct sembuf op[3] = {{0, 0, 0}, {0, 0, 0}, {0, 1, 0}};  is = 0;
  n = atoi(argv[1]);  op[0].sem_op = -n;   op[1].sem_op = n;
  mid = shmget(SHMKEY, n*sizeof(int), PERMS | IPC_CREAT);
  sid = semget(SEMKEY, 1, PERMS | IPC_CREAT);
  reg = shmat(mid, (char *) 0, 0);
  semctl(sid, 0, SETVAL, &is);
  numb = getpid();
  for (i=1 ; i<n ; i++) {
    pid = fork();
    if (!pid) {
      *((int *) reg + i - 1) = numb;
      semop(sid, &op[2], 1);  semop(sid, &op[0], 1);
      printf("%d %d\n", getpid(), *((int *) reg + i));
      semop(sid, &op[1], 1);  exit(0); }
    numb = pid; }
  *((int *) reg + n - 1) = numb;
  semop(sid, &op[2], 1);  semop(sid, &op[0], 1);
  printf("%d %d\n", getpid(), *((int *) reg));
  semop(sid, &op[1], 1);
  ..................................................... }
```

Επικοινωνία Διεργασιών

Δύο διεργασίες P_1 και P_2 επικοινωνούν μέσω υποδοχών ροής (stream sockets). Η διεργασία P_1 κάνει τρεις κλήσεις εγγραφής στην υποδοχή της S_1, σε απροσδιόριστες φάσεις της εκτέλεσής της, και γράφει, κατά σειρά, τα μηνύματα M_1, M_2 και M_3. Η διεργασία P_2 κάνει τρεις κλήσεις ανάγνωσης από την υποδοχή της S_2, τις R_1, R_2 και R_3, με τη σειρά αυτή, πάλι σε απρόβλεπτες χρονικές στιγμές. Ποια από τα παρακάτω ενδεχόμενα είναι πιθανό να συμβούν και ποια αποκλείονται παντελώς;

i. Η R_1 θα διαβάσει το μήνυμα M_1, η R_2 το M_2 και η R_3 το M_3.

ii. Η R_1 θα διαβάσει το μήνυμα M_3, η R_2 το M_1 και η R_3 το M_2.

iii. Η R_1 θα διαβάσει το μήνυμα M_1M_2 και η R_2 το M_3.

iv. Η R_2 θα διαβάσει το μήνυμα M_1M_3 και η R_3 το M_2.

v. Η R_1 θα διαβάσει το μήνυμα $M_1M_2M_3$.

vi. Η R_1 θα διαβάσει το μήνυμα M_1 και η R_2 το M_2M_3.

Δικαιολογήστε την απάντησή σας. Υπάρχουν άλλα πιθανά ενδεχόμενα, εκτός από αυτά που αναφέρονται και επιλέξατε; Ποιες θα ήταν οι απαντήσεις σας αν η επικοινωνία μεταξύ των διεργασιών γινόταν μέσω τηλεγραφικών υποδοχών (datagram sockets); Αν γινόταν μέσω σωλήνων;

Σήματα και Fork

Γράψτε ένα πρόγραμμα C (υποθέστε ότι ονομάζεται "sigcyc") που να καλείται με δύο ορίσματα $\langle N \rangle$ και $\langle M \rangle$ στη γραμμή εντολής. Η διεργασία-γονέας να δημιουργεί αρχικά $\langle N \rangle$ διεργασίες-παιδιά και στη συνέχεια να στέλνει ένα σήμα SIGUSR1 στο $\langle N \rangle$-οστό παιδί. Αυτό, όταν παραλάβει το σήμα, να στείλει επίσης ένα σήμα SIGUSR1 στο προηγούμενό του, το $\langle N \rangle - 1$ τάξης, αυτό στο προηγούμενό του και ούτω καθ' εξής. Το 1ο παιδί να στείλει το σήμα στο γονέα τους. Αυτή η κυκλική διάδοση του σήματος να γίνει συνολικά $\langle M \rangle$ φορές. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
$ ./sigcyc 3 2
Cycle 1: Process 0 (pid=11651) sending signal to 11654
Cycle 1: Process 3 (pid=11654) sending signal to 11653
Cycle 1: Process 2 (pid=11653) sending signal to 11652
Cycle 1: Process 1 (pid=11652) sending signal to 11651
Cycle 2: Process 0 (pid=11651) sending signal to 11654
Cycle 2: Process 3 (pid=11654) sending signal to 11653
Cycle 2: Process 2 (pid=11653) sending signal to 11652
Cycle 2: Process 1 (pid=11652) sending signal to 11651
```

Δυαδικοί Σηματοφόροι και Μεταβλητές Συνθήκης

Ποια ήταν, κατά τη γνώμη σας, η πρόθεση του προγραμματιστή που έγραψε το παρακάτω πρόγραμμα C; Παρότι το πρόγραμμα αυτό μεταγλωττίζεται με επιτυχία, έχει διάφορα σοβαρά λογικά λάθη. Γράψτε μία σωστή εκδοχή του, με βάση την απάντησή σας στην αρχική ερώτηση.

```
#include <stdio.h>
#include <pthread.h>
int i, j, count = 0;  pthread_t ker_id, pi_id;
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cvar = PTHREAD_COND_INITIALIZER;
void *kernighan(void *argp)
{  printf("I am Kernighan version %d\n", *(int *) argp);
   for (j=0 ; j<30000 ; j++) { pthread_mutex_lock(&mtx);  count++;
      if (count == 25000) {
         pthread_cond_signal(&cvar);  printf("%d: Signaled\n", ker_id); }
      pthread_mutex_unlock(&mtx); } }
void *pike(void *argp)
{  printf("I am Pike the only one\n");
   pthread_cond_wait(&cvar, &mtx);  printf("%d: Woke up\n", pi_id); }
main()
{  for (i=0 ; i<5 ; i++) pthread_create(&ker_id, NULL, kernighan, (void *) &i);
   pthread_create(&pi_id, NULL, pike, NULL);
   for (i=0 ; i<5 ; i++) pthread_join(ker_id, NULL);
   pthread_join(pi_id, NULL); }
```

Ουρές Μηνυμάτων

Γράψτε δύο προγράμματα C (έστω ότι ονομάζονται "filein" και "fileout") τέτοια ώστε το πρώτο να διαβάζει την προκαθορισμένη είσοδό του (stdin) και να την στέλνει στο δεύτερο μέσω μίας ουράς μηνυμάτων. Το πρόγραμμα "fileout" να στέλνει ό,τι διαβάζει από την ουρά μηνυμάτων στην προκαθορισμένη έξοδό του (stdout). Το κλειδί που θα αντιστοιχεί στην κοινή ουρά να είναι ο inode ενός αρχείου του οποίου το όνομα δίνεται σαν όρισμα και στα δύο προγράμματα. Πληροφοριακά, η κλήση συστήματος "msgrcv" επιστρέφει στο όνομά της, σε περίπτωση επιτυχούς εκτέλεσης, το πραγματικό μέγεθος του μηνύματος που διάβασε. Ενδεικτικές εκτελέσεις των δύο προγραμμάτων είναι οι εξής:

```
$ wc -c test_file
    104 test_file
$ cat test_file
This is a text file to test the
filein/fileout cooperation. Binary
files should be transmitted OK, too.
$ ./filein /etc/services < test_file
/etc/services inode is 11166
Referencing msgq with id 360448
Transferring data... Done!
Sent 104 bytes
$ wc -c filein
   6254 filein
$ ./filein /etc/fstab < filein
/etc/fstab inode is 4210
Referencing msgq with id 393217
Transferring data... Done!
Sent 6254 bytes
$
```

```
$ ./fileout /etc/services > outp_file
/etc/services inode is 11166
Referencing msgq with id 360448
Transferring data... Done!
Received 104 bytes
Removed msgq with id 360448
$ ./fileout /etc/fstab > new_filein
/etc/fstab inode is 4210
Referencing msgq with id 393217
Transferring data... Done!
Received 6254 bytes
Removed msgq with id 393217
$ wc -c outp_file
    104 outp_file
$ wc -c new_filein
   6254 new_filein
$ diff test_file outp_file
$ cmp filein new_filein
$
```

Βρες την ερώτηση/απάντηση

"Μπορούμε πριν το γράψιμο στο κατάλληλο άκρο ενός σωλήνα (ή σε μία υποδοχή ρωζ) του επιθυμητού μηνύματος, να γράφουμε το μέγεθος του μηνύματος (π.χ. σαν έναν ακέραιο 2 bytes), οπότε …". Αν το προηγούμενο είναι η αρχή μίας απάντησης, τότε ποια είναι η ερώτηση και πώς θα πρέπει να συμπληρωθεί η απάντηση;