ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Λειτουργικά Συστήματα

6ο εξάμηνο, Ακαδημαϊκή περίοδος 2023-2024

Άσκηση 3: Μηχανισμοί Εικονικής Μνήμης

1 Ασκήσεις

Στην παρούσα άσκηση θα ασχοληθείτε με τη μελέτη του μηχανισμού της εικονικής μνήμης και με τη χρήση του για αποδοτική διαδιεργασιακή επικοινωνία. Η άσκηση έχει δύο μέρη. Στο πρώτο μέρος σας ζητείται να πειραματιστείτε με κλήσεις συστήματος και να μελετήσετε βασικούς μηχανισμούς του ΛΣ (§ 1.1). Στο δεύτερο μέρος χρησιμοποιείτε την εικονική μνήμη για να μοιράσετε πόρους μεταξύ διεργασιών (processes) και να υλοποιήσετε διαδιεργασιακό σχήμα συγχρονισμού (§ 1.2). Προαιρετικά, μπορείτε να εφαρμόσετε το ίδιο και στην Άσκηση 1 (§ 1.3).

1.1 Κλήσεις συστήματος και βασικοί μηχανισμοί του ΛΣ για τη διαχείριση της εικονικής μνήμης (Virtual Memory – VM)

Στον φάκελο `/home/oslab/code/mmap` σας δίνεται ο σκελετός προγράμματος `mmap.c` τον οποίο και πρέπει να συμπληρώσετε ακολουθώντας τα εξής βήματα:

1. Τυπώστε το χάρτη της εικονικής μνήμης της τρέχουσας διεργασίας.
2. Με την κλήση συστήματος `mmap()` δεσμεύστε buffer (προσωρινή μνήμη) μεγέθους μίας σελίδας (page) και τυπώστε ξανά το χάρτη. Εντοπίστε στον χάρτη μνήμης τον χώρο εικονικών διευθύνσεων που δεσμεύσατε.

3. Βρείτε και τυπώστε τη φυσική διεύθυνση μνήμης στην οποία απεικονίζεται η εικονική διεύθυνση του buffer (τη διεύθυνση όπου βρίσκεται αποθηκευμένος στη φυσική κύρια μνήμη). Τι παρατηρείτε και γιατί;
4. Γεμίστε με μηδενικά τον buffer και επαναλάβετε το Βήμα 3. Ποια αλλαγή παρατηρείτε;
5. Χρησιμοποιείστε την `mmap()` για να απεικονίσετε (memory map) το αρχείο file.txt στον χώρο διευθύνσεων της διεργασίας σας και να τυπώσετε το περιεχόμενό του. Εντοπίστε τη νέα απεικόνιση (mapping) στον χάρτη μνήμης.
6. Χρησιμοποιείστε την `mmap()` για να δεσμεύσετε έναν νέο buffer, διαμοιραζόμενο (shared) αυτή τη φορά μεταξύ διεργασιών με μέγεθος μιας σελίδας. Εντοπίστε τη νέα απεικόνιση (mapping) στο χάρτη μνήμης.

Στο σημείο αυτό καλείται η συνάρτηση `fork()` και δημιουργείται μια νέα διεργασία.

7. Τυπώστε τους χάρτες της εικονικής μνήμης της γονικής διεργασίας και της διεργασίας παιδιού. Τι παρατηρείτε?
8. Βρείτε και τυπώστε τη φυσική διεύθυνση στη κύρια μνήμη του private buffer (Βήμα 3) για τις διεργασίες γονέα και παιδιού. Τι συμβαίνει αμέσως μετά το fork?
9. Γράψτε στον private buffer από τη διεργασία παιδί και επαναλάβετε το Βήμα 8. Τι αλλάζει και γιατί?
10. Γράψτε στον shared buffer (Βήμα 6) από τη διεργασία παιδί και τυπώστε τη φυσική του διεύθυνση για τις διεργασίες γονέα και παιδιού. Τι παρατηρείτε σε σύγκριση με τον private buffer?
11. Απαγορεύστε τις εγγραφές στον shared buffer για τη διεργασία παιδί. Εντοπίστε και τυπώστε την απεικόνιση του shared buffer στο χάρτη μνήμης των δύο διεργασιών για να επιβεβαιώσετε την απαγόρευση.
12. Αποδεσμεύστε όλους τους buffers στις δύο διεργασίες.

Δίδονται βοηθητικές συναρτήσεις στα αρχεία `/home/oslab/code/mmap/help.{c,h}`:

- `show_maps()`: Τυπώνει το χάρτη μνήμης της τρέχουσας διεργασίας. Περισσότερες πληροφορίες στο `man 5 proc` στο λήμμα `/proc/[pid]/maps`.
- `show_va_info()`: Τυπώνει πληροφορίες για την απεικόνιση στην οποία ανήκει μια συγκεκριμένη εικονική διεύθυνση.
- `get_physical_address()`: Εντοπίζει και επιστρέφει τη φυσική διεύθυνση στην οποία απεικονίζεται μια συγκεκριμένη εικονική διεύθυνση. Περισσότερες πληροφορίες στο `man 5 proc` στο λήμμα `/proc/[pid]/pagemap`.

Λεπτομέρειες σχετικά με τη χρήση της κλήσης συστήματος `mmap()`, π.χ. για τις σημαίες (flags), μπορείτε να βρείτε στο `man 2 mmap`.

1.2 Παράλληλος υπολογισμός Mandelbrot με διεργασίες αντί για νήματα

Ζητείται η τροποποίηση του προγράμματος υπολογισμού του Mandelbrot set, της άσκησης 2, ώστε αντί να χρησιμοποιεί threads (pthreads) για την παραλληλοποίηση του υπολογισμού του set, να χρησιμοποιεί διεργασίες (processes). Σας δίνεται σκελετός κώδικα που πρέπει να συμπληρώσετε/τροποποιήσετε στο `/home/oslab/code/sync-mmap`.

Η κατανομή του υπολογιστικού φόρτου γίνεται ανά σειρά, ακριβώς όπως στην περίπτωση των threads: Για n διεργασίες, η διεργασία i (με $i = 0, 1, 2, \ldots, n - 1$) αναλαμβάνει τις σειρές $i, i + n, i + 2 \times n, i + 3 \times n, \ldots$ Ο αριθμός των διεργασιών, `NPROCS`, όπως και ο αριθμός των threads στην προηγούμενη άσκηση, δίνεται ως όρισμα στη γραμμή εντολών.

1.2.1 Semaphores πάνω από διαμοιραζόμενη μνήμη

Ο συγχρονισμός μεταξύ των διεργασιών γίνεται και πάλι με semaphores, που παρέχονται από το πρότυπο POSIX, συμπεριλαμβάνοντας το αρχείο επικεφαλίδας <semaphore. h>. Δείτε περισσότερα στα manual pages `sem_overview(7)`, `sem_wait(3)`, `sem _post(3)`. Τα semaphores όμως θα πρέπει να βρίσκονται σε κοινή μνήμη στην οποία θα έχουν πρόσβαση όλες οι διεργασίες.

Για την διαχείριση της διαμοιραζόμενης μνήμης μεταξύ διεργασιών, σας δίνονται δύο βοηθητικές συναρτήσεις (`create_shared_memory_area`, `destroy_shared_memory_area`). Για την `create_shared_memory_area`, σας δίνεται ένας σκελετός, τον οποίο πρέπει να συμπληρώσετε, με χρήσης της `mmap()`, ώστε να δημιουργείται ένα memory mapping, το οποίο θα μπορεί να διαμοιραστεί μεταξύ των διεργασιών (anonymous shared memory mapping). Η συνάρτηση θα πρέπει να επιστρέφει την διεύθυνση (pointer) του mapping.

Ερωτήσεις:

1. Ποια από τις δύο παραλληλοποιημένες υλοποιήσεις (threads vs processes) περιμένετε να έχει καλύτερη επίδοση και γιατί; Πώς επηρεάζει την επίδοση της υλοποίησης με διεργασίες το γεγονός ότι τα semaphores βρίσκονται σε διαμοιραζόμενη μνήμη μεταξύ διεργασιών;
2. (Προαιρετική) Μπορεί το `mmap()` interface να χρησιμοποιηθεί για τον διαμοιρασμό μνήμης μεταξύ διεργασιών που δεν έχουν κοινό ancestor; Αν όχι, γιατί;

1.2.2 Υλοποίηση χωρίς semaphores

Ζητείται η τροποποίηση της υλοποίησης του προηγούμενου ερωτήματος, ώστε να αποφευχθεί η χρήση semaphores.

Οι διεργασίες-παιδιά υπολογίζουν τις γραμμές του set, όπως και πριν, αλλά αντί να τυπώνουν το αποτέλεσμα, το γράφουν στην αντίστοιχη γραμμή ενός διαμοιραζόμενο δισδιάστατου buffer, διαστάσεων `y_chars x x_chars`.
Η αρχική διεργασία αναλαμβάνει το output στο τερματικό, διατρέχοντας τις γραμμές του buffer, μετά την ολοκλήρωση του υπολογισμού από τις διεργασίες παιδιά.

1. Με ποιο τρόπο και σε ποιο σημείο επιτυγχάνεται ο συγχρονισμός σε αυτή την υλοποίηση; Πώς θα επηρεαζόταν το σχήμα συγχρονισμού αν ο buffer είχε διαστάσεις `NPROCS x x_chars`;

1.3 (Προαιρετική) Επέκταση Άσκησης 1

Στο 3ο ερώτημα της Άσκησης 1 (”Διαδιεργασιακή επικοινωνία”) δημιουργούσατε *P* διεργασίες παιδιά όπου η καθεμία μετρούσε πόσες φορές εμφανίζεται ένας χαρακτήρας σε ένα κομμάτι του αρχείου εισόδου. Κάθε διεργασία παιδί διατηρούσε το σύνολο των εμφανίσεων του χαρακτήρα σε μία τοπική μεταβλητή και επικοινωνούσε το τελικό αποτέλεσμα μέσω pipe στην γονική διεργασία. Εδώ σας ζητείται να τροποποιήσετε κατάλληλα το πρόγραμμά σας ώστε πλέον οι διεργασίες παιδιά να χρησιμοποιούν μία κοινή μεταβλητή (πάνω απο διαμοιραζόμενη μνήμη) για τη μέτρηση των εμφανίσεων του χαρακτήρα οι προσβάσεις στην οποία θα πρέπει να συγχρονίζονται με έναν semaphore πάνω απο διαμοιραζόμενη μνήμη, ομοίως με το ερώτημα 1.2.1 της παρούσας άσκησης. Με αυτόν τον τρόπο πλέον όταν το πρόγραμμά σας θα δέχεται Control+C απο το πληκτρολόγιο (δηλαδή το σήμα `SIGINT`) θα μπορεί να τυπώνει και την τρέχουσα τιμή του κοινού μετρητή, δηλαδή το πόσες εμφανίσεις του χαρακτήρα έχουν εντοπιστεί μέχρι τη συγκεκριμένη χρονική στιγμή.

2 Αναφορά άσκησης

Η προθεσμία για την εξέταση της άσκησης θα ανακοινωθεί στη σελίδα του μαθήματος. Μετά την εξέταση της άσκησης η κάθε ομάδα θα πρέπει να συντάξει μια (σύντομη) αναφορά και να την υποβάλλει στη σελίδα του μαθήματος. Η προθεσμία για την αναφορά είναι μια εβδομάδα μετά την προθεσμία εξέτασης της άσκησης.
Η αναφορά αυτή θα περιέχει:

- Τον πηγαίο κώδικα (source code) των ασκήσεων.
- Την έξοδο εκτέλεσης των προγραμμάτων.
- Σύντομες απαντήσεις στις ερωτήσεις.