Εκφώνηση Άσκησης 1

Διαχείριση Διεργασιών και Είσοδος/Εξοδος σε Αρχεία με Κλήσεις Συστήματος

Να υλοποιηθεί ένα πρόγραμμα σε **γλώσσα C** για το λειτουργικό σύστημα **Linux**, το οποίο θα εκτελεί τα εξής:

1. Ανάγνωση ορίσματος

- Το πρόγραμμα θα διαβάζει **έναν ακέραιο αριθμό N** ως όρισμα γραμμής εντολών κατά την εκτέλεσή του.
- Αν δεν δοθεί έγκυρος αριθμός, θα εμφανίζεται κατάλληλο μήνυμα σφάλματος και το πρόγραμμα θα τερματίζει.

2. Δημιουργία διεργασιών

- Η γονική διεργασία (**parent**) θα δημιουργεί **N διεργασίες-παιδιά** (**child processes**) χρησιμοποιώντας την κλήση συστήματος `fork()`.
- Θα αποθηκεύει τα **PIDs** των διεργασιών-παιδιών σε **πίνακα** που θα έχει δημιουργήσει με χρήση `malloc()`.

3. Ενέργειες κάθε διεργασίας παιδί

Κάθε παιδί, μόλις δημιουργηθεί, θα εκτελεί τα εξής:

- Θα εμφανίζει στην οθόνη ένα μήνυμα με το **PID του**.
- Θα δημιουργεί ένα αρχείο με όνομα `output_PID.txt`, όπου `PID` είναι το αναγνωριστικό της διεργασίας του.
- Θα γράφει στο αρχείο ένα μήνυμα της μορφής:
- `Child PID: <PID>, Parent PID: <PPID>`

  χρησιμοποιώντας **μόνο κλήσεις συστήματος** (`write()`).

- Θα τερματίζει την εκτέλεσή του.

4. Ενέργειες της γονικής διεργασίας

Μετά τη δημιουργία όλων των παιδιών, η γονική διεργασία:

- Θα περιμένει να ολοκληρωθούν όλες οι παιδικές διεργασίες χρησιμοποιώντας `wait()`.
- Στη συνέχεια, θα διαβάζει τα περιεχόμενα των αρχείων `output_PID.txt` που δημιουργήθηκαν από τα παιδιά, χρησιμοποιώντας **μόνο κλήσεις συστήματος** (`read()`).
- Θα εμφανίζει τα μηνύματα των παιδιών στην οθόνη.

5. Απαιτήσεις Υλοποίησης

- Όλες οι λειτουργίες εισόδου/εξόδου σε αρχεία πρέπει να γίνονται **αποκλειστικά με κλήσεις συστήματος** (`open()`, `write()`, `read()`, `close()`).
- Η δυναμική διαχείριση μνήμης πρέπει να γίνεται σωστά (`malloc()`, `free()`).
- Πρέπει να γίνεται **χειρισμός σφαλμάτων** για όλες τις κλήσεις συστήματος (π.χ., έλεγχος αποτυχίας `fork()`, `open()`, `write()`, `read()`, `malloc()` κ.λπ.).
- Ο κώδικας πρέπει να είναι **καλά δομημένος** και **καθαρός**, με κατάλληλα σχόλια όπου χρειάζεται.

6. Παράδειγμα Εκτέλεσης

Αν εκτελέσουμε την εντολή:

```
./process_manager 3
```

η έξοδος μπορεί να είναι:

```
Child process created: PID = 4621
Child process created: PID = 4622
Child process created: PID = 4623

Parent reading file contents:
Child PID: 4621, Parent PID: 4618
Child PID: 4622, Parent PID: 4618
Child PID: 4623, Parent PID: 4618
```

και θα έχουν δημιουργηθεί τα αρχεία:

- `output_4621.txt`
- `output_4622.txt`
- `output_4623.txt`