Temat zajęć	Procesy w systemie Linux
Zakres materiału	Tworzenie i wykonywanie procesów w systemie Linux

Materiał teoretyczny

- funkcje *fork(), wait(), exec...()*
- analiza argumentów linii poleceń
- analiza kodu zakończenia procesu

Treść zadania

Napisać języku C program spełniający poniższe wymagania:

- program akceptuje dokładnie jeden argument wywołania i jest nim ciąg cyfr dziesiętnych o długości od 1 do 20; uruchomienie programu z inną liczbą argumentów, albo bez argumentów, albo z argumentem dłuższym niż 20 znaków, albo z argumentem, który zawiera inne znaki niż cyfry dziesiętne powinno skutkować wyprowadzeniem na *stderr* informacji o błędzie i zakończeniem pracy z kodem zakończenia równym 1;
- jeżeli otrzymany argument ma długość 1, program kończy pracę, zwracając jako kod zakończenia wartość otrzymanej cyfry (np. dla `'1'` zwraca `1`, itd).
- w przeciwnym przypadku program dzieli otrzymany w argumencie łańcuch na dwie „połowy” (jeśli długość łańcucha nie jest parzysta, druga z „połówek” będzie dłuższa o jeden znak);
- program uruchamia dwa procesy potomne (funkcja *fork()*);
- każde z dzieci zastępuje swój kod (wybrana funkcja z rodziny *exec..()*, np. *execv()* albo *execl()*), uruchamiając ten sam program, który jest wykonywany przez rodzica i przekazuje mu jako argument odpowiednio: do pierwszego dziecka – pierwszą połowę łańcucha argumentu, do drugiego dziecka – drugą połowę argumentu;
- rodzic czeka na zakończenie obu procesów potomnych, a następnie wypisuje na *stdout*: swój PID, PID zakończonego dziecka, argument, z którym dziecko zostało uruchomione i kod zakończenia dziecka;
- rodzic kończy pracę, zwracając kod powrotu równy sumie kodów powrotów otrzymanych od dzieci

Uwaga: poprawna sekwencja operacji to *fork-fork-wait-wait* (czyli rodzic **najpierw** uruchamia **dwa** procesy potomne, a potem czeka na zakończenie obu); konstrukcja kodu, w której operacje układają się w ciąg *fork-wait-fork-wait* (procesy potomne pracują po kolei) dyskwalifikuje rozwiązanie!

- przykładowe wyjście z programu:

```
$ ./lab8 98765
 13533  13535                  9 9
 13533  13536                  8 8
 13532  13533                 98 17
 13534  13537                  7 7
 13538  13539                  6 6
 13538  13540                  5 5
 13534  13538                 65 11
 13532  13534                765 18
```

Uwaga! Kod źródłowy programu (1 plik) po oddaniu prowadzącemu zajęcia laboratoryjne musi zostać jako **załącznik** przesłany na adres `sos1@wi.zut.edu.pl`:

- plik z kodem źródłowym musi mieć nazwę: `numer_indeksu.so.lab08.c` (np. `66666.so.lab08.c`),
- plik musi zostać wysłany z poczty uczelnianej (domena `zut.edu.pl`),
- temat maila musi mieć postać: `SO IS1 999X LAB08`
  gdzie *999X* to numer grupy laboratoryjnej (np. `SO IS1 210C LAB08`),
  - w pierwszych trzech liniach skryptu w komentarzach (każda linia komentowana osobno) musi znaleźć się:
    - informacja identyczna z zamieszczoną w temacie maila,
    - imię i nazwisko osoby wysyłającej maila,
    - adres e-mail, z którego wysłano wiadomość

    np.:
    ```
    # SO IS1 210C LAB08
    # Jan Nowak
    # nj66666@zut.edu.pl
    ```
  - e-mail nie może zawierać żadnej treści (tylko załącznik).

Dostarczone kody źródłowe będą analizowane pod kątem wykrywania plagiatów. Niewysłanie wiadomości, wysłanie jej w formie niezgodnej z powyższymi wymaganiami lub wysłanie pliku, który nie będzie się poprawnie uruchamiał, będzie traktowane jako brak programu i skutkowało otrzymaniem za niego oceny niedostatecznej.