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

Materiał teoretyczny

- funkcje *fork(), wait()*, funkcje z rodziny *exec..(),* argumenty linii poleceń i kod zakończenia procesu

Treść zadania

Napisz w języku C program spełniający poniższe wymagania:

- program akceptuje dokładnie jeden argument wywołania – uruchomienie go z inną liczbą argumentów albo bez argumentów ma skutkować wyprowadzeniem na *stderr* informacji o błędzie i przerwaniem pracy z kodem zakończenia równym 11;
- otrzymany argument ma być ciągiem cyfr dziesiętnych – jeśli tak nie jest, program wyprowadza na *stderr* informację o błędzie i kończy pracę z kodem zakończenia równym 12;
- długość otrzymanego argumentu musi być mniejsza równa 40 – jeśli tak nie jest, program wyprowadza na *stderr* informację o błędzie i kończy pracę z kodem zakończenia równym 13;
- jeśli długość otrzymanego argumentu jest równa 1, program od razu kończy pracę z kodem zakończenia równym wartości otrzymanej cyfry, np. uruchomiony z argumentem „2” zwraca kod powrotu 2;
- jeśli długość otrzymanego argumentu jest równa 2, program od razu kończy pracę z kodem zakończenia równym wartości większej z otrzymanych cyfr, np. uruchomiony z argumentem „12” zwraca kod powrotu 2;
- program dzieli otrzymany argument na dwie równe części, a jeśli argument ma nieparzystą długość, na dwie części, z których druga jest o jeden znak dłuższa od pierwszej;
- 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 – lewą część otrzymanego argumentu, do drugiego dziecka – prawą część otrzymanego argumentu;
- rodzic czeka na zakończenie obu procesów potomnych, a następnie wypisuje na *stdout* trzy linie tekstu:
  1. swój PID, PID pierwszego zakończonego dziecka, argument, z którym pierwsze dziecko zostało uruchomione i kod powrotu pierwszego dziecka;
  2. swój PID, PID drugiego zakończonego dziecka, argument, z którym drugie dziecko zostało uruchomione i kod powrotu drugiego dziecka;
  3. swój PID i większy z kodów powrotów obu dzieci (wartość tę należy wyprowadzić w tej samej kolumnie, co kody powrotu potomków);

- rodzic kończy pracę, zwracając kod powrotu równy większemu z kodów powrotu obu dzieci;
- w efekcie w ostatniej linii wyjścia powinna znaleźć się wartość największej cyfry z łańcucha przekazanego do programu jako argument.

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:

```
./zad5 1234567
 17078  17080                  1 1
 17078  17081                 23 3
 17078                        3

 17079  17082                 45 5
 17079  17083                 67 7
 17079                        7

 17077  17078                123 3
 17077  17079               4567 7
 17077                        7
```

Uwaga! Kod źródłowy programu (1 plik) po zaprezentowaniu prowadzącemu zajęcia laboratoryjne musi zostać jako **załącznik** przesłany na adres `son1@wi.zut.edu.pl`:

- plik z kodem źródłowym musi mieć nazwę: `numer_indeksu.so.lab05.c` (np. `66666.so.lab05.c`),
- plik musi zostać wysłany z poczty uczelnianej (domena `zut.edu.pl`),
- temat maila musi mieć postać: `SO IN1 99X LAB05`
  gdzie *99X* to numer grupy laboratoryjnej (np. `SO IN1 20B LAB05`),
- w pierwszych trzech liniach kodu źródłowego 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 IN1 20B LAB05
// Jan Nowak
// nj66666@zut.edu.pl
```

- e-mail nie może zawierać żadnej treści (tylko załącznik).

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