Ustalenie notacji

Przez Σ^* oznaczmy zbiór słów nad alfabetem Σ. Dla słowa $w \in \Sigma^*$ przez $|w|$ oznaczmy jego długość. Dla $1 \leq i \leq |w|$ p
i-tą literę słowa w.

Deterministyczny automat skończony to piątka $\langle Q, \Sigma, \delta, q_0, F \rangle$, gdzie

- Q to zbiór stanów automatu;
- Σ to alfabet;
- $\delta : Q \times \Sigma \to Q$ to funkcja przejścia;
- $q_0 \in Q$ to stan początkowy;
- $F \subseteq Q$ to zbiór stanów akceptujących.

Automat akceptuje słowo $w \in \Sigma^*$, jeśli istnieje taki ciąg stanów $p_0, p_1, \ldots, p_{|w|}$, że $p_0 = q_0$, $p_{|w|} \in F$ oraz dla każdego $0 \leq i \leq |w| - 1$ zachodzi $\delta(p_i, w[i+1]) = p_{i+1}$.

Zbiór słów akceptowanych przez automat $\mathcal{A}$ oznaczamy przez $L(\mathcal{A})$ i nazywamy językiem rozpoznawanym przez automa

Opis urządzenia

Zadanie polega na napisaniu sterownika do urządzenia znakowego `/dev/dfa`, które symuluje deterministyczny automat sk
stanów Q i alfabetem Σ równymi $\{0,1\}^8$ (co odpowiada zakresowi typu `char`) oraz stanem początkowym $q_0 = 0$.

Opercja odczytu z urządzenia daje informację o tym, czy wczytane dotąd słowo należy do języka rozpoznawanego przez a
ujmując, próba odczytania n bajtów daje n bajtów równych 89, jeśli słowo należy do języka, lub n bajtów równych 78 w prz

Operacja zapisu pozwala na wprowadzenie kolejnych bajtów rozpoznawanego słowa.

Oprócz obsługi operacji czytania i pisania sterownik powinien implementować również operację `ioctl` pozwalającą na wyk
komend:

- DFAIOCRESET – resetuje aktualnie czytane słowo (automat wraca do stanu q_0);
- DFAIOCADD – przekazuje strukturę typu `char[3]` odpowiadającą trójce $\langle p, a, p' \rangle$ i zmienia aktualną funkcję przejścia au
  daną równianiem

  $$\delta'(q, c) = \begin{cases} p', \text{ jeśli } \langle q, c \rangle = \langle p, a \rangle, \\ \delta(q, c) \text{ w p. p.} \end{cases}$$

  Ponadto to polecenie resetuje aktualnie czytane słowo, tak jak powyżej.
- DFAIOCACCEPT – przekazuje zmienną typu `char` odpowiadającą stanowi p i modyfikuje zbiór stanów akceptujących aut
  do niego stanu p. To polecenie nie resetuje aktualnie czytanego słowa. Nie ma znaczenia, czy p należało do zbioru stan
  wywołaniem polecenia, w szczególności nie powinno to powodować błędu.
- DFAIOCREJECT – działaja podobnie jak DFAIOCACCEPT, z tym że przekazany stan p staje się stanem nieakceptującym.

Początkowo przyjmujemy $\delta(q, c) = 0$ dla dowolnych $q \in Q$ i $c \in \Sigma$ oraz $F = \emptyset$.

Urządzenie powinno zachowywać aktualną konfigurację (funkcja przejścia, stany akceptujące, informacje związane z doty
słowem) w przypadku uruchomienia `service update`.

Rozwiązanie zadania

Rozwiązanie powinno składać się z jednego pliku `dfa.c`, który będzie zawierał implementację urządzenia znakowego. Plik
przez nas `Makefile` zostanie umieszczony w katalogu `/usr/src/minix/drivers/dfa/`. Ponadto w katalogach `/usr/include/s
`/include/sys/` zostanie umieszczony plik `ioc_dfa.h`. Z kolei w `/etc/system.conf` zostanie umieszczony poniższy wpis:

```
service dfa
{
        system
                IRQCTL          # 19
                DEVIO           # 21
        ;
        ipc
                SYSTEM pm rs tty ds vm vfs
                pci inet lwip amddev
                ;
    uid 0;
};
```

Sterownik będzie kompilowany za pomocą dostarczonego przez nas `Makefile`, analogicznego jak dla przykładowego stero
`/usr/src/minix/drivers/dfa` zostaną wykonane polecenia:

```
make clean
make
make install

service up /service/dfa
service update /service/dfa
service down dfa
```

Oprócz poprawności rozwiązania na jego ocenę wpływ będzie miała również jego wydajność.

Oddawanie zadania

Rozwiązanie należy oddawać przez Moodle'a.

Ewentualne pytania również należy zadawać poprzez udostępnione tam forum.

Przykład

Zakładamy, że w katalogu, w którym się znajdujemy, jest plik wykonywalny `ioc_example`, którego kod źródłowy dołączamy c
on w ten sposób, że automat akceptuje napisy nieparzystej długości.

```
# mknod /dev/dfa c 20 0
# service up /service/dfa -dev /dev/dfa
# echo -n "Hello" > /dev/dfa
# head -c 4 /dev/dfa | xargs echo
NNNN
# ./ioc_example
# echo -n "Hello" > /dev/dfa
# head -c 2 /dev/dfa | xargs echo
YY
# echo -n "Hello" > /dev/dfa
# head -c 3 /dev/dfa | xargs echo
NNN…
```