✍ تمرین نهم os

«به نام خدا»

امیرمحمد پیرحسینلو
9531014

①

need

R_1 R_2

5 ∅

~~0 5~~

~~0 5~~

0 3 $\Longrightarrow$ $x = \emptyset$ $\Longrightarrow$ 0 3

$$\begin{array}{r} 0\ 3 \\ +\ 1\ 1 \\ \hline 1\ 4 \end{array} \Longrightarrow \text{deadlock}$$

~~9 5~~

$x = 5 \Longrightarrow$

$$\begin{array}{r} 5\ 3 \\ +\ 0\ 2 \\ \hline 5\ 5 \end{array}$$

با ترتیب P_5 و P_2 و P_3 و P_4 و $P1$ و $x = 5$ ، بن بست نداریم.

$x = 5 \Longleftarrow$

② deadlock پیش نمی آید زیرا هر فیلسوف (process) یا تمام resource های مورد نیاز خود را بدست می آورد یا هیچ چیزی بدست نمی آورد. به عبارتی allocation مربوط به آن یا برابر MAX می شود یا صفر.

بردار بردار

$\Longleftarrow$ بردار need برابر صفر می شود و process کار خود را جلو می برد. برای سایر process ها هم همچین اتفاقی می افتد. پس deadlock نداریم.

گرسنگی ممکن است پیش آید زیرا ممکن است یک process به طور مداوم چنگال ها را برداره، استفاده کنده، رها کند و دوباره برداره استفاده کند و به همین ترتیب

NEGARESTAN

$\Longleftarrow$ starvation ممکن است.

③ الف)

		allocation	Max	available	need
		ABCD	ABCD	ABCD	ABCD
①	P_1	0012	0012	1520 0012	0000
④	P_2	1000	1750	1532	0750
②	P_3	1354	2356	1354 2886	1002
③	P_4	0632	0652	0632 ② ⑭ ⑪ ⑧	0020
⑤	P_5	0014	0656		0642

به ترتیب P_1, P_3, P_4, P_2, P_5 اجرا کنیم $\Leftarrow$ deadlock نخواهیم داشت $\Leftarrow$ safe

ب) سیستم قادر به پاسخگویی است.

$$\begin{array}{r} 1520 \\ 0420 \\ \hline 1100 \end{array} \longrightarrow ✓$$

Date: ..../......./.......

الف)

4) هر منبعی درخواست داد، درخواستش را اجابت می کنیم چون m تا منبع داریم و هر پردازه نهایتا m منبع می خواهد.
حال صبر می کنیم تا کار این پردازه تمام شود و منابع آزاد شود.
حال درخواست پردازه بعدی را اجابت می کنیم و آن هم شبیه همین درخواست اول است و مشکلی پیش نمی آید
به همین ترتیب پیش رویم مشکلی نخواهد داشت.

ب) در بدترین حالت ~~یک~~ همه پردازه ها جز یکی، به یک منبع نیاز دارند و طبیعتا آن یک پردازه به $m+n-1-n = m-1$ منبع نیاز خواهد داشت. ($m-1$ بیشترین نیاز یک پردازه است). حال چون m تا منبع داریم، در بدترین حالت نوبت به پردازه با نیاز $m-1$ منبع می شود که می توانیم درخواست آن را اجابت کنیم.
پس از پایان کار این پردازه، $m-1$ منبع آزاد می شود و همچنان m منبع داریم. چون سایر پردازه ها کمتر از m منبع نیاز دارند، می توانیم آن ها را هم اجرا کنیم بدون مشکل $\Leftarrow$ deadlock نخواهیم داشت.

Date: / /

```
monitor problem{                                                    (5)
    enum {noneed; using ; waiting} state [n];
    condition self[n];
    void use (ink i){
            state[i] = waiting;
            test(i);
            if( state [i] != using)
                    self [i] .wait();
    }

    void release (int i){
            state [i] = noneed;
            test((i+n-1) %n);
            test ((i+1) %n);
    }

    void test (int i) {
          if((state [(i+n-1) %n] != using) &&
             (state[i] == waiting) && (state[(i+1)
              %n] != using)) {
              state[i] = using;
              self [i]. signal;
          }
}
initializtion() {
       for (int i=0; i<5; i++)
}           state[i] = noneed;
```

Date: ....../....../......

0 1 2 3 4 5 6 7 8 9 10 11 12 13