Arkadiusz Chomik (ID205)
Konsultacja naukowa,
rozwiązanie części zadań
oraz sprawdzenie poprawności

Jan Kurnatowski (ID202)
Opracowanie elektroniczne,
rozwiązanie części zadań
oraz sprawdzenie poprawności

Wersja dokumentu: 1.1 (03.02.2003 godzina 17:45)

Zadanie

Dana jest mapa bitowa systemu plików, w którym blok ma 4kB, a fragment 1 kB. Poniżej mapy podane są adresy początkowe kolejnych fragmentów. Należy zaadresować plik o rozmiarze 15kB. Należy podać (w wolnym wierszu) nowy stan mapy bitowej, a także ile i które wskaźniki adresowe i-węzła będą wykorzystane do zaadresowania pliku i jakie adresy będą w nich umieszczone? Odpowiedź uzasadnij.

Numeracja fragmentów i bloków od 0.

Mapa bitowa zajętości poszczególnych bloków i fragmentów

fragmenty	zajęte:	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	numery:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
numery bloków:		0				1				2				3				4				5			

Mapa bitowa opisuje, który z fragmentów jest zajęty (w odpowiedniej komórce wartość logiczna 1), a który pusty (0).
W treści zadania otrzymaliśmy następujące dane:

$$RozmiarBloku = 4kB$$
$$RozmiarFragmentu = 1kB$$
$$RozmiarPliku = 15kB$$

Rozwiązujemy stosując poniższe wzory:

$$LiczbaFragmentówBloku = \frac{RozmiarBloku}{RozmiarFragmentu}$$

$$LiczbaFragmentówBloku = \frac{4kB}{1kB} = 4$$

Sprawdzamy ile wolnych bloków (żaden fragment bloku nie może być zajęty) musimy poświęcić na zapisanie naszego pliku. Korzystamy ze wzoru:

$$LiczbaBlokówPliku = \left\lfloor \frac{RozmiarPliku}{RozmiarBloku} \right\rfloor$$

$$LiczbaBlokówPliku = \left\lfloor \frac{15kB}{4kB} \right\rfloor = 3$$

Sprawdzamy ile fragmentów pozostało nam jeszcze do zapisania (ostatnie fragmenty pliku):

$$LiczbaFragmentówPliku = \left\lceil \frac{RozmiarPliku - LiczbaBlokówPliku \cdot RozmiarBloku}{RozmiarFragmentu} \right\rceil$$

$$LiczbaFragmentówPliku = \left\lceil \frac{15kB - 3 \cdot 4kB}{1kB} \right\rceil = 3$$

Potrzebujemy do zapisania naszego pliku 3 wolne bloki i 3 wolne fragmenty. Przeglądając mapę bitową od lewej do prawej wybieramy trzy kolejne wolne bloki (nie muszą ze sobą sąsiadować) i wpisujemy do nich plik. Następnie wybieramy dowolny pusty lub częściowo zajęty blok i wpisujemy do niego sekwencyjnie pozostałe fragmenty. Jedyny warunek jest następujący: w wybranym bloku musi być wystarczająco dużo sąsiadujących ze sobą wolnych fragmentów. W poniższej mapie bitowej pogrubioną czcionką zaznaczono zajęte fragmenty oraz pierwsze fragmenty zapisanych bloków. Na czerwono fragmenty, w których nie można wpisać ostatnich fragmentów pliku (bo w bloku 0 są tylko dwa wolne fragmenty, a potrzebujemy miejsca na trzy fragmenty pliku).

Mapa bitowa zajętości poszczególnych bloków i fragmentów

fragmenty	zajęte:	1	1	0	0	1	0	0	1	**1**	**1**	**1**	**1**	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	numery:	0	1	2	3	4	5	6	7	**8**	9	10	11	12	**13**	14	15	**16**	17	18	19	**20**	21	22	23
numery bloków:		0				1				2				3				4				5			

Kolejne bloki pliku zapisaliśmy do bloków 2, 4 i 5. Jako adres bloku bierzemy adres pierwszego fragmentu w tym bloku. Pozostałe ostatnie 3 fragmenty pliku zapisujemy w bloku 3 zaczynając od zapisania danych w fragmencie 13. Informacje te zapisujemy w i-węźle.

i-węzeł pliku		**Zajęte fragmenty na dysku**	**Fragmenty pliku na dysku**
numer wskaźnika adresowego i-węzła	adres początkowy bloku lub ciągu fragmentów	które fragmenty pliku zostały zapisane w których fragmentach na dysku	
1	8	8,9,10,11	0,1,2,3
2	16	16,17,18,19	4,5,6,7
3	20	20,21,22,23	8,9,10,11
4	13	13,14,15	12,13,14

Zadanie

O godzinie 16.10 wydano następujące polecenia:

$ date > /tmp/plikA ; ln /tmp/plikA /tmp/plikB

następnie o godz. 18.00 wydano polecenie

$ ln /tmp/plikA /tmp/plikC; cat /tmp/plikC

a o godz.20.20 wydano polecenie

$ date >> /tmp/plikB

Podaj dokładnie, co znajduje się w i-węźle pliku /tmp/plikC (Uwaga: przyjąć że wyjście polecenia date zajmuje 30 bajtów). Podaj uzasadnienie.

Rodzaj informacji	Informacja	Uwagi
typ pliku:	zwykły	Domyślnie plik jest zwykły
prawa dostępu do pliku:	rw- r-- ---	Dowolny, należy jednak pamiętać że żaden utworzony plik nie ma ustawianych automatycznie praw wykonywania (ani dla właściciela, ani dla grupy, ani dla reszty użytkowników)
liczba dowiązań:	3	Nazwy plików przechowuje się w plikach katalogów a nie w i-węźle. Dowiązań jest tyle, ile dany plik ma dowiązanych nazw. Zaraz po utworzeniu plik ma dowiązaną jedną nazwę (czyli jedno dowiązanie). Każde dowiązanie miękkie (ln) zwiększa liczbę dowiązań, bo zwiększa się liczba nazw związanych z danym plikiem. Czyli wszystkie trzy „pliki": plikA, plikB i plikC mają wspólny i-węzeł oraz wspólny plik na dysku.
id właściciela:	zuza	Dowolny, można tutaj podać swój login
id grupy:	grp_zuzy	Dowolny, można podać swoją grupę unixową
rozmiar pliku w bajtach:	60B	Trzeba zsumować, ile razy nastąpiło dopisanie danych do pliku. W tym przykładzie przy tworzeniu pliku plikA (16:10) wpisano od razu do niego tekst zwracany przez polecenie date (30B) – użyto do tego celu polecenia date > nazwa_pliku. O godzinie 20:20 dopisano do pliku plikB (a tym samym do pliku plikA, bo w rzeczywistości istnieje na dysku tylko jeden plik o trzech nazwach: plikA, plikB i plikC) po raz kolejny datę (30B) – użyto polecenia date >> nazwa_pliku. Czyli ostatecznie plik zawiera dwie daty: jedną z godziny 16:10 i drugą z 20:20.
czas ostatniej modyfikacji pliku:	20:20	Czas modyfikacji może być późniejszy od czasu dostępu dlatego, że modyfikując plik nie musimy go odczytywać (np. wtedy, gdy dopisujemy coś do pliku – np. date >> /tmp/plikB).
czas ostatniego dostępu do pliku:	18:00	Jest to czas ostatniego odczytu zawartości pliku. Może być wcześniejszy od czasu modyfikacji pliku. Użyto

		polecenia cat tmp/plikC (wyświetlenie na ekran)
czas ostatniej zmiany w i-węźle:	20:20	Wszystkie zmiany w którymkolwiek z wierszy tej tabeli powodują aktualizację tego czasu. Nie powodują aktualizacji zmiany takie jak np. zmiana nazwy pliku (bo nazwa pliku zapisana jest w pliku katalogu, nie w i-węźle). O tej porze nastąpiło dopisanie do pliku nowych danych, więc zmieniła się informacja o rozmiarze pliku oraz czas ostatniej aktualizacji pliku. Mogła ulec zmiana liczby wskaźników adresowych pliku, ale w tym przypadku ona nie wystąpiła.
liczba wskaźników adresowych pliku:	1	Plik ma 60B. Możemy założyć, że blok ma np. 4kB a fragment 1kB. Czyli plik mieści się w jednym fragmencie. Adres tego fragmentu jest przechowywany w jednym wskaźniku.

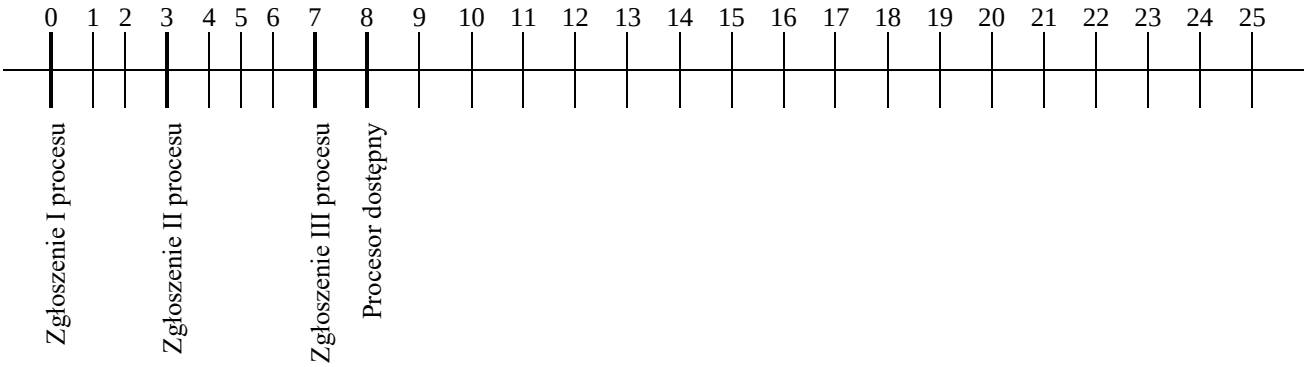
Zadanie

W chwili 00h 00m 00s zostaje zgłoszone zadanie użytkownika i utworzony proces 1, trzy sekundy później zostaje utworzony proces 2 i jeszcze cztery sekundy później proces 3. Przewidywany czas wykonania procesu 1 wynosi 5 sek, procesu 2 wynosi 50 s, a procesu 3 wynosi 2 sek. Procesor jest dostępny do przetwarzania tych procesów od chwili 00h 00m. 08s. Czas przełączania kontekstu proszę pominąć. Policzyć średni czas przetwarzania tych procesów dla algorytmu FCFS. Należy podać sposób rozwiązania i uzasadnienie.

Czas przełączania kontekstu (pominięty) to czas potrzebny systemowi do przełączenia się między procesami.

Zaznaczamy na osi czasu poszczególne czasy zgłoszeń procesów oraz chwilę, od której procesor jest dostępny do przetwarzania tych procesów.
Proces I zgłoszony został w chwili 0s. Proces II – 3s. Proces III – 7s. Procesor jest dostępny do przetwarzania procesów od chwili 8s. Proces I będzie wykonywany przez 5s (procesor łącznie poświęci na wykonywanie procesu I 5s swego czasu), II – 50s, III – 2s.

Oś czasu (w sekundach)

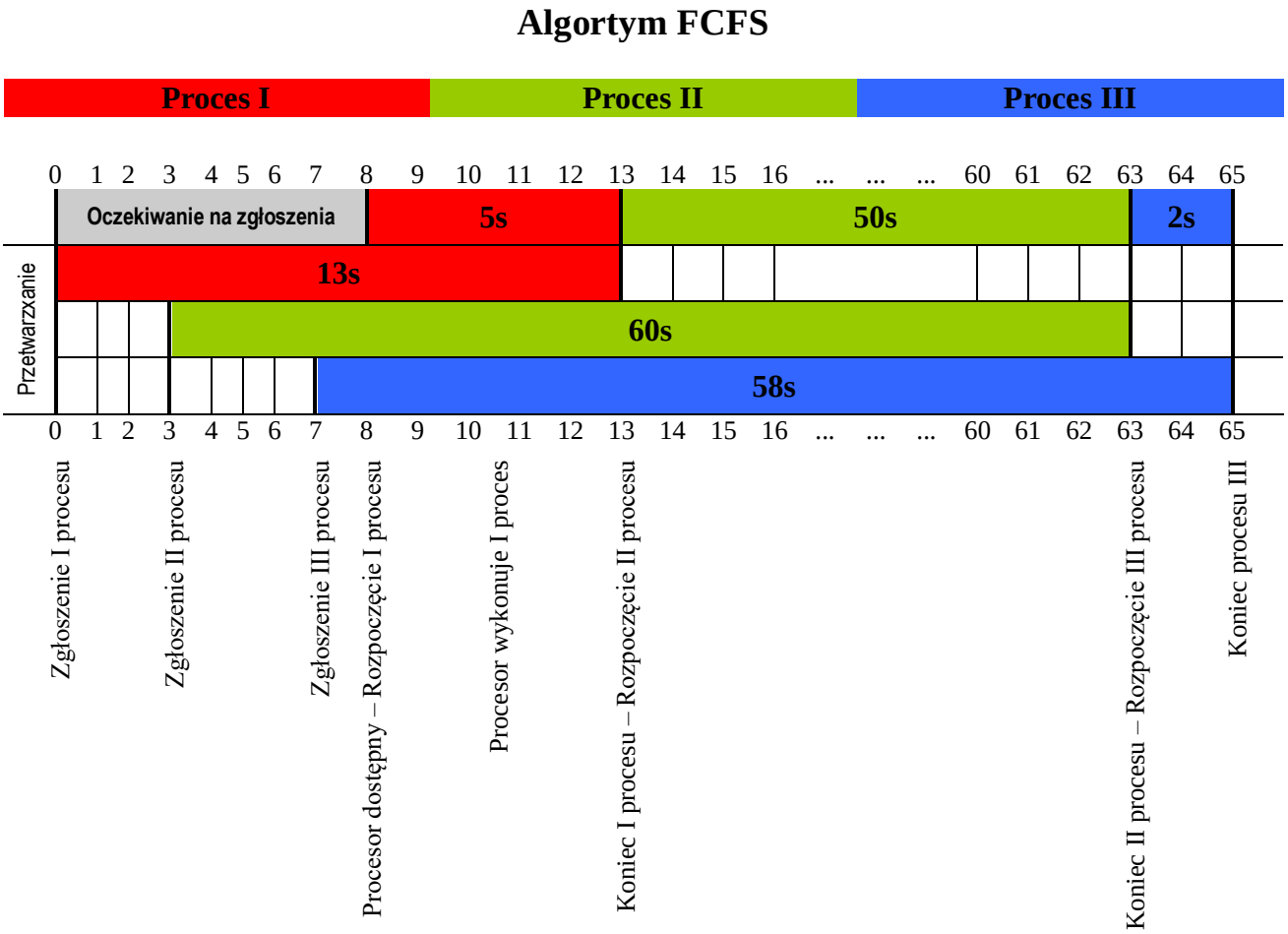


$$Czas\ \Pr zetwarzania\ \Pr ocesu = CzasZakonczenia\ \Pr ocesu - CzasZgloszenia\ \Pr ocesu$$

$$Czas\ \Pr zetwarzania\ \Pr ocesu \geq CzasWykonywania\ \Pr ocesu$$

$$\acute{S}redniCzas\ \Pr zetwarzania\ \Pr ocesu = \acute{S}redniaArytmetycznaCzasów\ \Pr zetwarzaniaPoszczególn ych\ \Pr ocesów$$

Algorytm FCFS (First Come First Served) – procesy obsługiwane są w całości w kolejności ich zgłaszania.



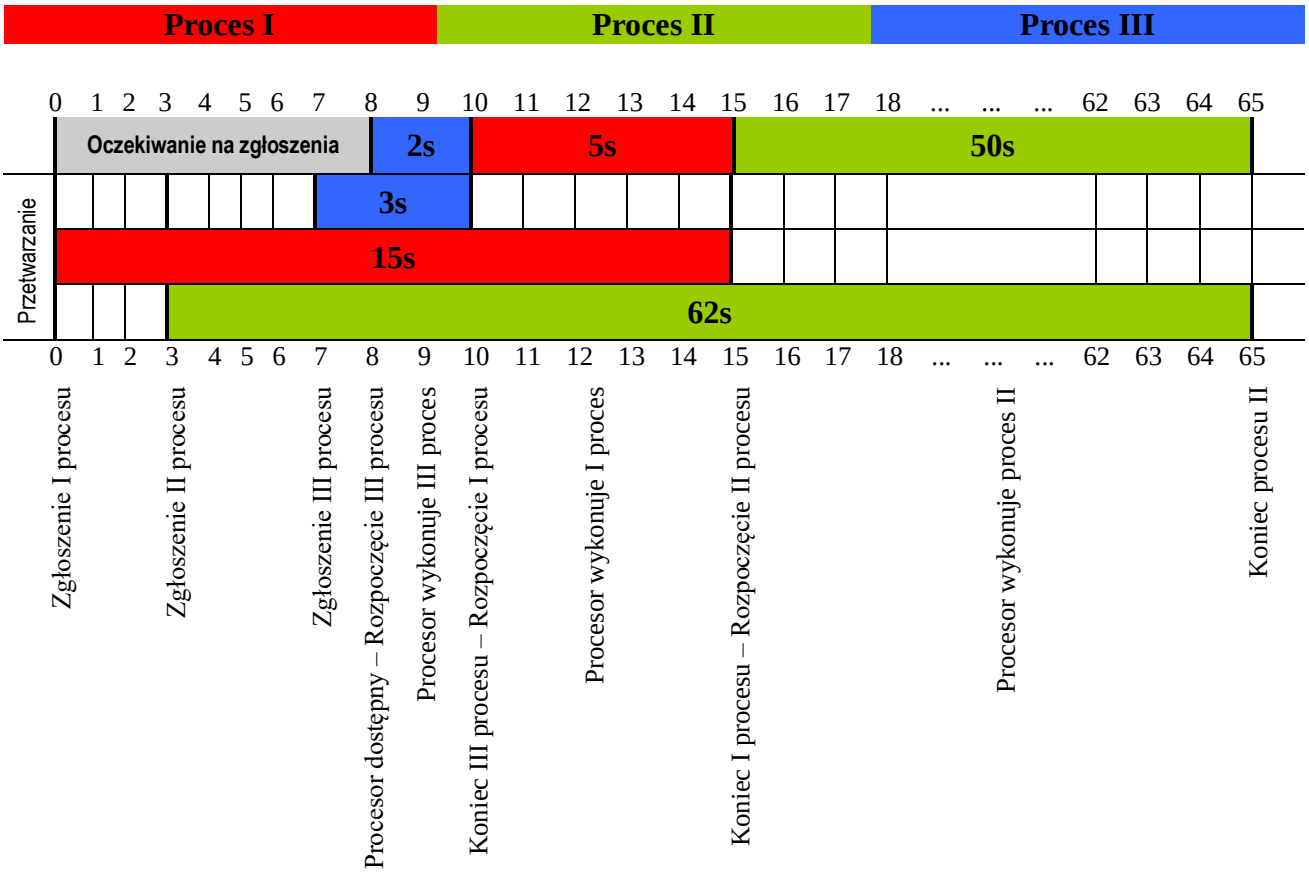
$$CzasZakonczenia\ \Pr ocesuI = 8s + 5s = 13s$$
$$CzasZakonczenia\ \Pr ocesuII = 13s + 50s = 63s$$
$$CzasZakonczenia\ \Pr ocesuIII = 63s + 2s = 65s$$

$$Czas\ \Pr zetwarzania\ \Pr ocesuI = 13s - 0s = 13s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuII = 63s - 3s = 60s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuIII = 65s - 7s = 58s$$

$$\acute{S}redniCzas\ \Pr zetwarzania\ \Pr ocesu = \frac{13s + 60s + 58s}{3} \approx 42{,}67\,\mathrm{s}$$

Algorytm SJF (Short Job First) – sortuje procesy niemalejąco wzlędem ich długości (czasu wykonywania), a następnie działa podobnie jak algorytm FCFS.

Algortym SJF



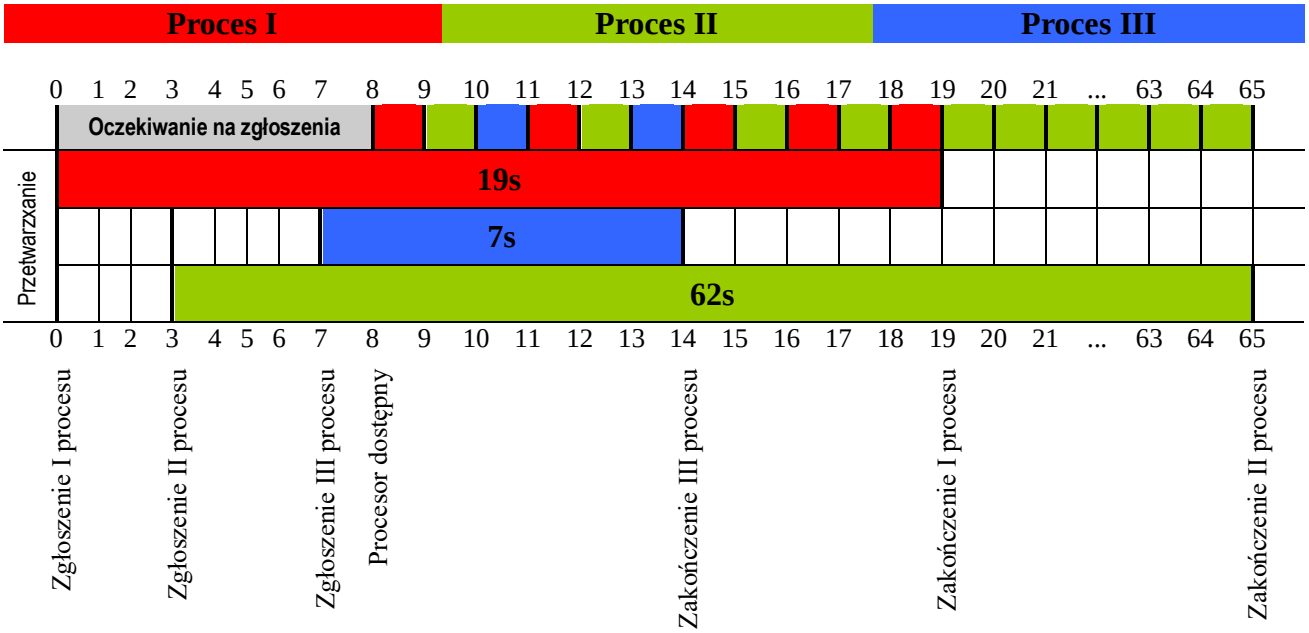
$$CzasZakonczenia\ \Pr ocesuI = 10s + 5s = 15s$$
$$CzasZakonczenia\ \Pr ocesuII = 15s + 50s = 65s$$
$$CzasZakonczenia\ \Pr ocesuIII = 8s + 2s = 10s$$

$$Czas\ \Pr zetwarzania\ \Pr ocesuI = 15s - 0s = 15s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuII = 65s - 3s = 62s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuIII = 10s - 7s = 3s$$

$$\acute{S}redniCzas\ \Pr zetwarzania\ \Pr ocesu = \frac{15s + 62s + 3s}{3} \approx 26{,}67s$$

Algorytm rotacyjny – procesor wykonuje procesy cyklicznie, poświęcając jednorazowo każdemu z nich pewną jednostkę czasu, zwaną kwantem czasu. Stosowany w systemach z podziałem czasu.

Algortym Rotacyjny



$$KwantCzasu = 1s$$

$$CzasZakonczenia\ \Pr ocesuI = 19s$$
$$CzasZakonczenia\ \Pr ocesuII = 65s$$
$$CzasZakonczenia\ \Pr ocesuIII = 14s$$

$$Czas\ \Pr zetwarzania\ \Pr ocesuI = 19s - 0s = 19s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuII = 65s - 3s = 62s$$
$$Czas\ \Pr zetwarzania\ \Pr ocesuIII = 14s - 7s = 7s$$

$$\acute{S}redniCzas\ \Pr zetwarzania\ \Pr ocesu = \frac{19s + 62s + 7s}{3} \approx 29{,}33s$$

Zadanie

Jeden blok ma rozmiar 4kB. Jaki jest maksymalny rozmiar pliku w tym systemie. Założenie, że wskaźnik zajmuje 4B.

Mamy 12 wskaźników adresowania bezpośredniego, 1 wskaźnik adresowania pośredniego pojedynczego, 1 wskaźnik adresowania podwójnego i 1 wskaźnik adresowania potrójnego.

$$AdresowaneBezposrednie = RozmiarBloku \cdot 12$$

$$LiczbaWskaźnikówWBloku = \frac{RozmiarBloku}{RozmiarWskaznika}$$

$$AdresowaniePojedyncze = LiczbaWskaźnikówWBloku \cdot RozmiarBloku$$

$$AdresowaniePodwojne = LiczbaWskaźnikówWBloku^2 \cdot RozmiarBloku$$

$$AdresowaniePotrójne = LiczbaWskaźnikówWBloku^3 \cdot RozmiarBloku$$

$$NajwiekszyRozmiarPliku = AdresowanieBezposrednie + AdresowaniePojedyncze + \\ + AdresowaniePodwojne + AdresowaniePotrójne$$

W tym przykładzie:

$$AdresowaneBezposrednie = 4kB \cdot 12 = 48kB$$

$$LiczbaWskaźnikówWBloku = \frac{4kB}{4B} = \frac{4 \cdot 1024B}{4B} = 1024$$

$$AdresowaniePojedyncze = 1024 \cdot 4kB = 4MB$$

$$AdresowaniePodwojne = 1024^2 \cdot 4kB = 4GB$$

$$AdresowaniePotrójne = 1024^3 \cdot 4kB = 4TB$$

$$NajwiekszyRozmiarPliku = 48kB + 4MB + 4GB + 4TB \approx 4TB$$

Zadanie

Dany zbiór roboczy trzech ramek o numerach: 6, 7, 8. Proces ma 8 stron numerowanych od 0 do 7. Stosując algorytm LRU podaj kolejne zawartości tablicy stron procesu oraz kolejne zawartości stosu błędów stron. Podano odwołania do stron i stan wskaźników czasowych:

Odwołanie	Czas
3	11
4	15
5	17
4	18
7	20
6	22

Stan wyjściowy tablicy stron procesu:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	-	n	-
4	-	n	-
5	-	n	-
6	-	n	-
7	-	n	-

Czyli żadna z 8 stron procesu nie jest w żadnej z przydzielonych procesowi ramek. Tablica informuje nas, czy dana strona jest w ramce. Jeśli strona jest w ramce, wskaźnik czasowy informuje nas, kiedy strona była ostatnio z tej ramki czytana. Bit poprawności ustawiony jest na n, gdy strona nie jest wczytana do żadnej z ramek, w przeciwnym wypadku p.

W czasie 11 przyszło odwołanie do strony 3. Sprawdzamy, czy strona 3 jest zapisana w którejś z ramek. Nie jest, więc przydzielamy tej stronie pierwszą wolną ramkę – 6. System ładuje stronę 3 do ramki 6. Uzupełniamy wiersz dotyczący strony 3 odpowiednimi informacjami:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	**6**	**p**	**11**
4	-	n	-
5	-	n	-
6	-	n	-
7	-	n	-

W czasie 15 przyszło odwołanie do strony 4. Sprawdzamy, czy strona 4 jest zapisana w którejś z ramek. Nie jest, więc przydzielamy tej stronie pierwszą wolną ramkę – 7. System

ładuje stronę 4 do ramki 7. Uzupełniamy wiersz dotyczący strony 4 odpowiednimi informacjami:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	6	p	11
4	**7**	**p**	**15**
5	-	n	-
6	-	n	-
7	-	n	-

W czasie 17 przyszło odwołanie do strony 5. Sprawdzamy, czy strona 5 jest zapisana w którejś z ramek. Nie jest, więc przydzielamy tej stronie pierwszą wolną ramkę – 8. System ładuje stronę 5 do ramki 8. Uzupełniamy wiersz dotyczący strony 8 odpowiednimi informacjami:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	6	p	11
4	7	p	15
5	**8**	**p**	**17**
6	-	n	-
7	-	n	-

W czasie 18 przyszło odwołanie do strony 4. Sprawdzamy, czy strona 4 jest zapisana w którejś z ramek. Tak, strona 4 zapisana jest w ramce 7, więc aktualizujemy w wierszu dotyczącym strony 4 informacje o czasie ostatniego odczytu strony.

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	6	p	11
4	7	p	**18**
5	8	p	17
6	-	n	-
7	-	n	-

W czasie 20 przyszło odwołanie do strony 7. Sprawdzamy, czy strona 7 jest zapisana w którejś z ramek. Nie, więc szukamy pierwszej wolnej ramki, ale takiej nie ma, bo wszystkie trzy ramki są zajęte. Trzeba jedną z nich zwolnić. Stosujemy algorytm LRU (Least Recently Used) – zastąp stronę, która nie była używana od najdłuższego czasu. W naszym przypadku

jest to strona 3 w ramce 6. Do ramki 6 wpisujemy stronę 7 i aktualizujemy dane dla wiersza przechowującego informacje o stronie 7 i o stronie 3:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	**-**	**n**	**-**
4	7	p	18
5	8	p	17
6	-	n	-
7	**6**	**p**	**20**

W czasie 22 przyszło odwołanie do strony 6. Sprawdzamy, czy strona 6 jest zapisana w którejś z ramek. Nie, więc szukamy pierwszej wolnej ramki, ale takiej nie ma, bo wszystkie trzy ramki są zajęte. Trzeba jedną z nich zwolnić. Stosujemy algorytm LRU (Least Recently Used) – zastąp stronę, która nie była używana od najdłuższego czasu. W naszym przypadku jest to strona 5 w ramce 8. Do ramki 8 wpisujemy stronę 6 i aktualizujemy dane w wierszu przechowującym informacje o stronie 6 i o stronie 5:

strona	ramka	bit	czas
0	-	n	-
1	-	n	-
2	-	n	-
3	-	n	-
4	7	p	18
5	**-**	**n**	**-**
6	**8**	**p**	**22**
7	6	p	20

Błąd został wygenerowany przez odwołania do stron 3, 4, 5, 7, 6 (bo nie było ich w ramkach w chwili, gdy zostało wysłane żądanie odczytania danej strony). Tworzymy stos reprezentujący błędy stron. Stos ma zawsze wysokość równą liczbie ramek. Na początku stos jest pusty. Numer strony nie generującej błędu (w naszym wypadku strona 4) wypływa w stosie na wierzch. Na stosie od dołu (spód stosu) do góry (wierzch stosu) poukładane są numery stron załadowanych do ramek w kolejności odpowiednio od najdawniej do ostatnio używanej strony (sortowanie rosnące według wskaźników czasowych). Gdy przychodzi odwołanie do strony 7, której nie ma w żadnej z ramek (patrz wyżej) usuwana jest ze stosu strona która była odczytywana najdawniej (strona 3), a na wierzch stosu odkładany jest numer strony 7.

Odwołania i wskaźniki czasowe								
Strony		3	4	5	4	7	6	
Czas		*11*	*15*	*17*	*18*	*20*	*22*	
Stan stosu (tylko dane wytłuszczone)				**5** *(17)*	**4** *(18)*	**7** *(20)*	**6** *(22)*	strona ostatnio odczytana
			4 *(15)*	**4** *(15)*	**5** *(17)*	**4** *(18)*	**7** *(20)*	(w nawiasach czas ostatniego odczytania strony)
		3 *(11)*	**3** *(11)*	**3** *(11)*	**3** *(11)*	**5** *(17)*	**4** *(18)*	strona odczytana najdawniej

Zadanie

W trakcie wykonywania pewnego 560B programu zaobserwowano następujący ciąg odwołań do pamięci (podane liczby wskazują adresy komórek pamięci, nie są to numery stron):
120, 231, 173, 309, 185, 245, 176, 46, 434, 458, 364, 550, 520
Wyznacz ciąg odwołań do stron zakładając, że rozmiar strony wynosi 100B, a programowi przydzielono obszar pamięci 200B.

$$LiczbaRamek = \frac{RozmiarObszaruPamieci}{RozmiarStrony}$$

$$LiczbaRamek = \frac{200B}{100B} = 2$$

Programowi przydzielono dwie ramki po 100B.

Musimy podzielić program na strony:

Podział programu na strony

Strona	Adresy bajtów programu
0	0-99
1	100-199
2	200-299
3	300-399
4	400-499
5	500-559

Przyporządkowujemy odwołaniom do konkretnych bajtów programu odpowiednie strony, na których te bajty można znaleźć:

Odwołania do bajtów i odpowiadających im stron programu

Bajt	120	231	173	309	185	245	176	46	434	458	364	550	520
Strona	1	2	1	3	1	2	1	0	4	4	3	5	5

Zadanie

Rozważając następujący układ odwołań do stron:

1, 4, 2, 1, 3, 5, 6, 1, 2, 1, 2, 6, 3, 7, 6, 3, 2, 1, 2, 4, 1

i przyjmując, że do dyspozycji mamy trzy ramki, wyznacz liczbę błędów braku strony dla algorytmu „Optymalnego”. (Uwaga: na początku wszystkie ramki były puste, tzn pierwsze odwołania do stron zawsze generują błąd strony). Podać uzasadnienie.

Zbiór roboczy to zbiór stron procesu jednocześnie znajdujących się w pamięci (przechowywanych w udostępnionych procesowi ramkach). Gdy występuje odwołanie do strony, której nie ma w zbiorze roboczym (nie ma w żadnej z ramek), generowany jest błąd strony i żądana strona zapisana jest w określonej ramce (w zależności od zastosowanego algorytmu). Ramka to strona pomocnicza należąca do obszaru pamięci przydzielonego programowi przez system.

Trzy algorytmy:
FIFO – zastąp stronę, która była wprowadzona do obszaru roboczego (ramek) najwcześniej.
LRU (Least Recently Used) – zastąp stronę, która nie była używana od najdłuższego czasu (badanie ciągu ostatnich odwołań).
Optymalny – Zastąp stronę, która najdłużej nie będzie używana (badanie ciągu odwołań, które dopiero przyjdą). W praktyce najtrudniejsza implementacja.
Uwaga: czas zmiany zawartości ramki (wprowadzenia do niej strony) i czas ostatniego użycia ramki (odczytania z niej strony) to dwie całkowicie różne informacje!

W każdym z algorytmów na początku zachodzi sytuacja jak poniżej (Ø – ramka pusta):

Algorytm FIFO, LRU, Optymalny

Odwołanie		**1**	**4**	**2**	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	**1**	1	1																		
Ramka II:	Ø	Ø	**4**	4																		
Ramka III:	Ø	Ø	Ø	**2**																		
Błąd:		T	T	T																		
Numer:	**a**	**b**	**c**	**d**	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

a. Wszystkie ramki są puste
b. Odwołanie do strony 1. W żadnej z ramek nie ma takiej strony i istnieje co najmniej jedna ramka pusta. Wybieramy więc pierwszą w kolejności pustą ramkę (I) i zapisujemy do niej zawartość strony 1. Generowany jest błąd strony.
c. Odwołanie do strony 4. W żadnej z ramek nie ma takiej strony i istnieje co najmniej jedna ramka pusta. Wybieramy więc pierwszą w kolejności pustą ramkę (II) i zapisujemy do niej zawartość strony 4. Generowany jest błąd strony.
d. Odwołanie do strony 2. W żadnej z ramek nie ma takiej strony i istnieje co najmniej jedna ramka pusta. Wybieramy więc pierwszą w kolejności pustą ramkę (III) i zapisujemy do niej zawartość strony 2. Generowany jest błąd strony.

Na tym etapie wszystkie ramki są zajęte. Teraz zaczynają się rozbieżności między poszczególnymi algorytmami w sposobie wybierania ramki, do której zostanie skopiowana zawartość żądanej strony.

Algorytm FIFO:

e. Odwołanie do strony 1. Strona 1 jest w ramce I. Brak błędu. Nie odnotowujemy niczego w tabelce.

Algorytm FIFO

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1																		
Ramka II:	Ø	Ø	4	4																		
Ramka III:	Ø	Ø	Ø	2																		
Błąd:		T	T	T																		
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

f. Odwołanie do strony 3. Strony 3 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 3. Algorytm FIFO zastępuje zawartość ramki, która od najdłuższego czasu nie była zmieniana. W naszym wypadku jest to ramka I, gdyż strona 1 siedzi w niej od najdłuższego czasu.

Algorytm FIFO

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		3																
Ramka II:	Ø	Ø	4	4		4																
Ramka III:	Ø	Ø	Ø	2		2																
Błąd:		T	T	T		T																
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

g. Odwołanie do strony 5. Strony 5 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 5. Algorytm FIFO zastępuje zawartość ramki, która od najdłuższego czasu nie była zmieniana. W naszym wypadku jest to ramka II, gdyż strona 4 siedzi w niej od najdłuższego czasu.

Algorytm FIFO

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		3	3															
Ramka II:	Ø	Ø	4	4		4	5															
Ramka III:	Ø	Ø	Ø	2		2	2															
Błąd:		T	T	T		T	T															
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

Ostatecznie powstaje dla algorytmu FIFO następująca tabelka:

Algorytm FIFO

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	**1**	1	1		**3**	3	3	**1**	1				1	**7**	7		7	**1**		1	
Ramka II:	Ø	Ø	**4**	4		4	**5**	5	5	**2**				2	2	**6**		6	6		**4**	
Ramka III:	Ø	Ø	Ø	**2**		2	2	**6**	6	6				**3**	3	3		**2**	2		2	
Błąd:		T	T	T		T	T	T	T	T				T	T	T		T	T		T	
Numer:	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**r**	**s**	**t**	**u**	**w**	**v**

Liczba błędów stron: 14.

Algorytm LRU

e. Odwołanie do strony 1. Strona 1 jest w ramce I. Brak błędu. Nie odnotowujemy niczego w tabelce.

Algorytm LRU

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1																		
Ramka II:	Ø	Ø	4	4																		
Ramka III:	Ø	Ø	Ø	2																		
Błąd:		T	T	T																		
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

f. Odwołanie do strony 3. Strony 3 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 3. Algorytm LRU zastępuje zawartość tej ramki, która nie była używana (odczytywana) od najdłuższego czasu. W ramce I przechowywana jest strona 1, ostatnio odczytywana w punkcie e. W ramce II przechowywana jest strona 4 ostatnio odczytywana w punkcie c. W ramce III przechowywana jest strona 2 ostatnio odoczytywana w punkcie d. Tak więc najdłużej nie była używana ramka II. Wprowadzamy więc do ramki II zawartość strony 3.

Algorytm LRU

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1																
Ramka II:	Ø	Ø	4	4		3																
Ramka III:	Ø	Ø	Ø	2		2																
Błąd:		T	T	T		T																
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

g. Odwołanie do strony 5. Strony 5 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 5. Algorytm LRU zastępuje zawartość tej ramki, która nie była używana (odczytywana) od najdłuższego czasu. W ramce I przechowywana jest strona 1, ostatnio odczytywana w punkcie e. W ramce II przechowywana jest strona 3 ostatnio odczytywana w punkcie f. W ramce III przechowywana jest strona 2 ostatnio odoczytywana w punkcie d. Tak więc najdłużej nie była używana ramka III. Wprowadzamy więc do ramki III zawartość strony 5.

Algorytm LRU

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1	1															
Ramka II:	Ø	Ø	4	4		3	3															
Ramka III:	Ø	Ø	Ø	2		2	5															
Błąd:		T	T	T		T	T															
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

w. Odwołanie do strony 4. Postępujemy podobnie jak w poprzednich punktach.

Algorytm LRU

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1	1	6	6	6				6	6			6	1		1	
Ramka II:	Ø	Ø	4	4		3	3	3	1	1				3	3			3	3		4	
Ramka III:	Ø	Ø	Ø	2		2	5	5	5	2				2	7			2	2		2	
Błąd:		T	T	T		T	T															
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

Jak widać, te odwołania które były jasnoszare stają się szare, a odwołania szare stają się ciemnoszare. Strony ciemnoszare to te, które nie były odczytywane od najdłuższego czasu. Ramka przechowująca taką stronę jest używana do wpisania strony aktualnie żądanej do odczytu.

Ostatecznie powstaje następująca tabelka:

Algorytm LRU

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1	1	6	6	6				6	6			6	1		1	
Ramka II:	Ø	Ø	4	4		3	3	3	1	1				3	3			3	3		4	
Ramka III:	Ø	Ø	Ø	2		2	5	5	5	2				2	7			2	2		2	
Błąd:		T	T	T		T	T	T	T	T				T	T			T	T		T	
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

Liczba błędów stron: 13

Algorytm Optymalny:

e. Odwołanie do strony 1. Strona 1 jest w ramce I. Brak błędu. Nie odnotowujemy niczego w tabelce.

Algorytm Optymalny

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1																		
Ramka II:	Ø	Ø	4	4																		
Ramka III:	Ø	Ø	Ø	2																		
Błąd:		T	T	T																		
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

f. Odwołanie do strony 3. Strony 3 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 3. Algorytm Optymalny zastępuje zawartość ramki, która przez najbliższy czas najdłużej będzie nieużywana (sprawdzanie odwołań z przyszłości). Strona 4 z ramki II będzie wykorzystywana dopiero w punkcie w. Więc wybieramy ramkę II i wpisujemy do niej stronę 3.

Algorytm Optymalny

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1																
Ramka II:	Ø	Ø	4	4		3																
Ramka III:	Ø	Ø	Ø	2		2																
Błąd:		T	T	T		T																
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

g. Odwołanie do strony 5. Strony 5 nie ma w żadnej z trzech ramek. Generowany jest błąd strony. Musimy więc wybrać jedną ramkę i przepisać do niej zawartość strony 5. Algorytm Optymalny zastępuje zawartość ramki, która przez najbliższy czas najdłużej będzie nieużywana (sprawdzanie odwołań z przyszłości). Strona 3 z ramki II będzie wykorzystywana dopiero w punkcie n. Wybieramy więc ramkę II i wpisujemy do niej stronę 5.

Algorytm Optymalny

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	1	1	1		1	1															
Ramka II:	Ø	Ø	4	4		3	5															
Ramka III:	Ø	Ø	Ø	2		2	2															
Błąd:		T	T	T		T	T															
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

Algorytm Optymalny

Odwołanie		1	4	2	1	3	5	6	1	2	1	2	6	3	7	6	3	2	1	2	4	1
Ramka I:	Ø	**1**	1	1		1	1	1						**3**	3			**2**	2		**4**	
Ramka II:	Ø	Ø	**4**	4		**3**	**5**	**6**						6	6			6	**1**		1	
Ramka III:	Ø	Ø	Ø	**2**		2	2	2						2	**7**			7	7		7	
Błąd:		T	T	T		T	T	T						T	T			T	T		T	
Numer:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	w	v

Liczba błędów stron: 11.