Wydział Informatyki PB,Egzamin z Systemów operacyjnych II 14.02.2005

Poprawne odpowidzi na zielono i z podkreśleniem

1. Co to jest panika jądra - ang. kernel panic (odpowiedź nie więcej niż trzy zdania) ?

   Jest to kontrolwane zawieszenie się jądra w reakcji na błąd jądra.

2. Bezpośrenim skutkiem wywołania funkcji wake_up(struct wait_queue **q) jest: (A) przyznanie procesora procesowi znajdującemu się na początku kolejki q; (B) wywołanie funkcji schedule(); (C) Czasami uśpienie procesu, w którym wystąpiło wywołanie funkcji wake_up; **(D)** Odpowiedzi A, B i C są nieprawidłowe.

   Komentarz: Obudzenie procesu w Linuksie nie jest związane z wywołaniem planisty. W przypadku funkcji wake_up następuje jedynie ustawienie stanu na TASK_RUNNING, dodanie do kolejki procesów gotowych (ang. runqueue) i usunięcie z kolejki q.

3. Plik zapisany w systemie plików ext2 jest otwarty do odczytu przez 3 procesy użytkownika user1, do zapisu 2 procesy użytkownika user2 oraz do odczytu przez 1 proces użytkownika root. Ile obiektów VFS i-węzła tego pliku znajduje się w pamięci?

   1, ponieważ i-węzły VFS są współdzielone (patrz algorytmy iget oraz iput)

4. W kolejce procesów gotowych znajdują się procesy: P1 – klasy SCHED_OTHER, P2 – klasy SCHED_RR z rt_priority == 80, P3 – klasy SCHED_FIFO z rt_priority == 80. Funkcja schedule została wywołana przez proces P4 klasy SCHED_RR z rt_priority == 81 i state==TASK_RUNNING. Który proces otrzyma procesor (zakładamy, że P1 jest pierwszy w kolejce)?

   P4, ponieważ ma stan TASK_RUNNING (a więc jest kandydatem do wyboru) i ma najwyższy priorytet.

5. Chcemy wyzerować pewien blok na dysku. Podaj jakie funkcje podsystemu pamięci buforowej i w jakiej kolejności należy wywołać aby zrobić to w sposób minimalizujący obciążenie dysku. Wystarczy podać tylko nazwy wywoływanych funkcji (ale we właściwej kolejności) oraz nie trzeba zajmować się zwalnianiem bufora przez brelse/bforget.

   getblk, mark_buffer_dirty, mark_buffer_uptodate.

6. Które urządzenia są reprezentowane przez pliki specjalne ? (A) klawiatura; (B) napęd cd; (C) karta sieciowa. Poprawny wybór to: AB (klawiatura jest częścią konsoli)

7. Urządzenie znakowe wykorzystuje tryb PIO i przerwania. Porzebuje bufora o długości 2MB. Jakich funkcji jądra wolno użyć do przydzielenia pamięci dla bufora tego urządzenia: (A) vmalloc (B) kmalloc (C) __get_free_pages. Poprawne funkcje to: A (kmalloc i get_free_pages mają limit około 128KB)

8. Dlaczego zastosowanie bufora TLB w procesorze zwiększa koszt przełączenia kontekstu? (max cztery zdania).

   Przełączenie kontekstu wymaga zmiany tablicy stron (rejestr CR3), co wiąże się z unieważnieniem (nie zapisem, jak pisała część z Państwa) bufora TLB. Sprawia to, że żądania translacji numeru strony na numer ramki przez pewien czas, do momentu ponownego wypełnienia TLB, muszą być zapokojone z tablicy stron z pamięci RAM, co spowalnia pracę procesora.

9. Dlaczego para funkcji cli()/sti(), blokujących i odblokujących przerwania, sama w sobie nie zapewniłaby ochrony spójności danych jądra w systemie wieloprocesorowym ? (max trzy zdania).

   cli blokuje przerwanie tylko na lokalnym procesorze. W systemie SMP przerwanie mógłby odebrać inny procesor.

10. Proces w systemie dwuprocesorowym SMP wykonuje się w dwóch wątkach, przy czym mamy gwarancję że każdy wątek wykonuje się na innym procesorze. Wątek pierwszy trylion razy czyta globalną zmienna X, a wątek drugi trylion razy czyta globalną zmienną Y. Poza odczytem swojej zmiennej (można przyjąć że do rejestru procesora) w pętli, wątki nic nie robią. Po zakonczeniu pętli wątki i cały proces kończą pracę. W przypadku P1 mamy gwarancję że zmienne X oraz Y znajdują się w tym samym wierszu pamięci podręcznej, a w przypadku P2 mamy gwarancję, że te zmienne znajdują się w różnych wierszach pamięci podręcznej. Można stwierdzić że: (A) w przypadku P1 proces wykona się w czasie istotnie szybszym niż w przypadku P2. (B) w przypadku P2 proces wykona się w czasie istotnie szybszym niż w przypadku P1. **(C)** nie będzie istotnych różnic w szybkości pomiędzy przypadkami P1 oraz P2 (D) W jednym z tych przypadków proces wykona się w czasie istotnie szybszym, ale nie wiadomo w któym

Komentarz: Każdy procesor wykonuje odczyt danych, który może zostać wykonany z lokalnej pamięci cache. Ponieważ nie ma zapisu, nie ma konieczności podejmowania akcji utrzymujących spójność współdzielonych wierszy.

11. Proces P1 klasy SCHED_OTHER wykonuje się w trybie jądra. Proces P2 klasy SCHED_FIFO oczekuje na odczyt jednego znaku z konsoli. Użytkownik naciska klawisz, co powoduje obudzenie procesu P2. Na skutek tego zdarzenia: (A) Proces P1 zostanie natychmiast wywłaszczony, a procesor otrzyma proces P2. (B) Zmienna state w strukturze opisującej proces P1 zostanie ustawiona na TASK_INTERRUPTIBLE. (C) Obydwie odpowiedzi A i B są prawidłowe. **(D)** Obydwie odpowiedzi A i B są nieprawidłowe.

Komentarz: Patrz pytanie 2

12. Pojedyncze żądanie przekazane do procedury strategii: **(A)** dotyczy zawsze ciągłego obszaru (blok kolejnych sektorów) dysku. (B) dotyczy zawsze ciągłego obszaru w pamięci (C) Obydwie odpowiedzi A i B są prawidłowe. (D) Obydwie odpowiedzi A i B są nieprawidłowe.

Komentarz: Żądanie może dotyczyć grupy bloków "rozrzuconych" po pamięci.

13. Wyjątek stronicowania może zostać wygenerowany (A) Przez proces działający w trybie jądra próbujący zapisać dane pod adresem uzyskanym w wyniku pomyślnego wywołania funkcji kmalloc() **(B)** Przez proces działający w trybie jądra, próbujący odczytać dane pod poprawnym adresem z przestrzeni adresowej procesu (C) Obydwie odpowiedzi A i B są prawidłowe (D) Obydwie odpowiedzi A i B są nieprawidłowe.

Komentarz: Pamięć procesu podlega wymianie. W związku z tym odwołanie do niej może wygenerować wyjątek np. w sytuacji gdy strona jest w obszarze wymiany i nie ma jej w pamięci. Są też inne sytuacje (np. copy on write, load on demand, zero on demand)

14. Które funkcje jądra mogą uśpić wołający je proces: (A) kmalloc(size, GFP_KERNEL) (B) kmalloc (size,GFP_ATOMIC), (C) put_user. Są to: AC

15. Jaka jest różnica pomiędzy funkcjami sleep_on a interruptible_sleep_on ? (jedno zdanie)

Proces uśpiony przez interruptible_sleep_on jest budzony przez sygnał, a przez sleep_on nie.

16. Wywołanie dolnych połów (ang. bottom half) może nastąpić w trakcie (A) powrotu z szybkiego przerwania (B) powrotu z wywołania systemowego (C) powrotu z wolnego przerwania (D) wejścia do jądra w wyniku wywołania systemowego przez proces. Poprawne odpowiedzi to: BC

17. Pamięć systemowa składa się z 8 ramek. Zgodnie ze strategią bliźniaków (ang. buddy), tak jak to robi Linux 2.0.x, przydzieliliśmy najpierw 5 ramek a później 2 ramki. Na końcu zwolniliśmy przydzielone 5 ramek. Podaj ile wolnych bloków i o jakim rozmiarze pozostanie w systemie buddy. Wynik podaj w postaci: rozmiar bloku (liczba ramek) – liczba takich bloków pozostała w systemie.

1 x 4 strony + 1 x 2 strony.

18. Wiadomo, że cała zawartość pliku, zapisanego na płycie umieszczonej w bardzo starym, wolnym napędzie cdrom pracującym w trybie PIO, znajduje się w podręcznej pamięci buforowej. **(A)** odczyt pliku za pomocą mmap będzie istotnie szybszy, niż odczyt za pomocą read (B) odczyt pliku za pomocą mmap będzie istotnie wolniejszy, niż odczyt za pomocą read (C) nie wiadomo, która metoda odczytu będzie istotnie szybsza. (D) ponieważ czas ewentualnego kopiowania w pamięci jest nieznaczący w porównaniu z czasem transmisji danych z tak powolnego urządzenia, nie będzie istotnych różnic w czasie odczytu.

Komentarz: Ponieważ wszystkie dane pliku są już w pamięci czas transmisji z urządzenia jest nieistotny, bo nie będzie żadnej transmisji. Natomiast odczyt przez mmap nie wymaga kopiowanie danych (strony z buforami są po prostu umieszczane w tablicy stron procesu) i dlatego jest szybszy od read, które wymaga conajmniej jednej kopii.

19. Proces w Linuksie 2.0.x na systemie x86 wywołuje jądro poprzez **(A)** przerwanie programowe int 80H (B) instrukcję syscall (C) instrukcję dalekiego skoku jmp (D) instrukcję dalekiego wywołania podprogramu call.

20. Zwolnienie bufora pakietu sk_buff przekazanego sterownikowi karty sieciowej do wysłania (A) następuje w metodzie hard_start_xmit sterownika (B) dokonywane jest przez jądro, gdy pakiet został już wysłany **(C)** dokonywane jest w handlerze obsługi przerwania karty (D) wszystkie odpowiedzi A-C są błędne.

Wydział Informatyki PB,Egzamin z Systemów operacyjnych II 14.02.2005
Imię i Nazwisko:

Na liście z wynikami proszę umieścić (zaznacz właściwy kwadracik) □ moje imię i nazwisko
□ mój numer indeksu, który jest równy .....

Punktacja: **Za każde pytanie testowe 2 pkt, za zadanie maksymalnie 30 pkt, 35pkt zalicza.**

Zadanie: Urządzenie może obsługiwać cztery porty równoległe (dla drukarki, w standardzie Centronics). Wysposażone jest w cztery porty danych o adresach DATA0, DATA1, DATA2, DATA3 i port stanu o adresie STAT. Urządzenie działa następująco: zapis do portu danych (jednego bajtu) powoduje rozpoczęcie przesyłania znaku do drukarki. Gdy przesłanie się powiedzie (albo gdy nastąpi błąd transmisji) następuje zgłoszenie przerwania. W przypadku pomyślnej operacji oznacza to, że dany port gotowy jest do przyjęcia kolejnego znaku (wysłanie następnego znaku, jeszcze przed potwierdzeniem przez przerwanie, jest błędem). Przyczyna zgłoszenia przerwania jest wskazywana przez rejestr STAT w sposób następujący: najmłodze dwa bity określają numer portu zgłaszającego przerwanie (np. 10 oznacza port DATA2), a najstarszy bit równy określa przyczynę przerwania (1 - pomyślne zakończenie transmisji znaku, 0 - błąd transmisji.) Napisz sterownik urządzenia, obsługujący cztery niezależne numery podrzędne, (każdy dla innego portu). Podaj implementację funkcji **open**, **release**, **write**, oraz funkcji **device_irq** implementującej handler przerwania. Implementacja funkcji rejestrujących i usuwających urządzenie oraz kodu rezerwującego przerwanie (wywołanie request_irq) nie jest potrzebna (zakładamy że przerwanie zostało już zarezerwowane i przydzielone funkcji **device_irq**). Ponadto zakładamy, że każdy port równoległy może być otwarty tylko jeden raz, a próba kolejnego otwarcia kończy się zwróceniem błędu -EBUSY (odpada problem synchronizacji procesów). **Wskazówka:** Dla tak wolnego urządzenia niezbędne jest zastosowanie mechanizmu zasypiania / budzenia, w którym proces zasypia w oczekiwaniu na zakończenie transmisji znaku, a jest budzony po zakończeniu tej transmisji przez handler przerwania. Należy również pamiętać o obsłudze sygnałów oraz o tym aby sterownik nie “zgubił przerwania” **Punktacja** Za rozwiązanie zadania, przy założeniu, że nie występują błędy transmisji można otrzymać 20 pkt. Za uwzględnienie błędów transmisji w sposób następujący: w przypadku błędu próbuj retransmisji tego samego znaku. Jeżeli trzy próby transmisji zakończą się niepowodzeniem, to wyjdź z funkcji write zwracając kod błędu można dodatkowo otrzymać 10 pkt.

Pomoc dla zadania:
current->signal – maska bitowa zgłoszonych sygnałów.
current->blocked – maska bitowa zablokowanych sygnałów.
-ERESTARTSYS – proces przerwany przez sygnał.
-ENODEV – nie ma takiego urządzenia.
-EIO - błąd transmisji wejścia/wyjścia
char inb(short port); // odczyt bajtu z portu
void outb(short port,char byte); // zapis bajtu do portu
char get_user(char *user_address);
void init_waitqueue(struct wait_queue ** p);
void sleep_on(struct wait_queue ** p);
void interruptible_sleep_on(struct wait_queue ** p);
void wake_up(struct wait_queue ** p);

Funkcje sterownika, które trzeba zaimplementować w rozwiązaniu zadania:
int open(struct inode *inode, struct file *file);
int release(struct inode *inode, struct file *file);
int write(struct inode *inode, struct file *file,char *buffer,int size);
void device_irq(int irq, void * dev_id, struct pt_regs *pt)

Numer podrzędny urządzenia: MINOR(inode->i_rdev);