**#Intrebare 1**

ls -l /proc | grep ^d | awk '{print $9}' | grep '^[0-9]\*$'

**#Intrebare 2**

Se vor afisa in consola toate numerele de la 00,01,02,03....pana la 99

**#Intrebare 3**

Acesta directiva contine 3 macro-uri: $(EXE),$(OBJS),$(LD)

De obicei intr-un makefile, $(OBJS) specifica toate fisierele obiect necesare pentru a construi programul principal. $LD face parte dintr-o serie predefinita de macrori si indica linker-ul utilizat. ($(CC)... CC=gcc) Si $(EXE) reprezinta referirea unui macro ce de va purta numele fisierului executabil creat.Prin $@ se specifica tinta curenta pentru make (in cazul nostru continutul macro-ului $(EXE).

Prima linie reprezinta o linie de dependente in care sunt precizate tintele si linia 2 reprezinta o linie de comenzi in care se compileaza tintele

**#Intrebare 4**

Practic acest apel realizeaza un nou proces, numit proces fiu, iar procesul care a apelat fork devine proces parinte a noului proces creat. Cele doua procese sunt identice in termeni de continut al segmentelor lor de date si text si aproape identice in ceea ce priveste segmentele sistem. Singurele diferente intre astfel de procese apar la nivelul unor atribute care trebuie sa fie diferite (cum ar fi PID care trebuie sa fie unic pentru fiecare proces în parte).

Procesul fiu va mosteni de la parinte toti descriptorii de fisier deschisi de catre acesta, asa ca orice prelucrari ulterioare in fisiere vor fi efectuate in punctul in care le-a lasat parintele.

Deoarece codul parintelui si codul fiului sunt identice si pentru ca aceste procese vor rula in continuare in paralel, trebuie facuta clar distinctia, in interiorul programului, intre actiunile ce vor fi executate de fiu si cele ale parintelui. Cu alte cuvinte, este nevoie de o metoda care sa indice care este portiunea de cod a parintelui si care a fiului.

Acest lucru se poate face simplu, folosind valoarea returnata de functia fork( ). Ea returneaza:

-1, daca operatia nu s-a putut efectua (eroare)

0, in codul fiului

pid, in codul parintelui, unde pid este identificatorul de proces al fiului nou-creat.

pid\_t fork(void);

**#Intrebare 5** - vezi problemaPunctul5.c

**#Intrebare 6**

Avand in vedere ca i nu este o variabila globala, inseamna ca in functia ce defineste threadul valoarea lui i nu poate fi schimbata. Deci la sfarsitul executiei secventei de cod i va avea valoarea 1 datorita incrementarii ++i.