1. Ariketa

a)

gcc -o kateluzapena kateluzapena.c /home/emmam/ikasi/azterketaAdib/fkateak.c -I /home/emmam/ikasi/azterketaAdib/include/

b)

make fitxategia begiratu

c)

gcc -I /home/emmam/ikasi/azterketaAdib/include/ -c fkateak.c -o fkateak.o

ar rv libkateak.a fkateak.o

ranlib libkateak.a

gcc -L /home/emmam/ikasi/azterketaAdib/ -I /home/emmam/ikasi/azterketaAdib/include/ -o kateluzapena2 kateluzapena.c -lkateak -static

2. Ariketa

2.1. Azaldu gainetik, programa nagusi honek egiten duena.

Bi estruktura sortzen ditu. Lehenengo 3 parametro sartu direla konprobatzen da, bestela errore mezua pantailaratuko da. Ondoren, lehenengo fitxategiaren i nodoaren datuak lehenengo estruktutan kargatzen ditu, -1 itzultzen badu, errorea gertatu da. x aldagaian inodoko zenbakia gordetzen du eta bigarren fitxategiarekin egiten du gauza berdina (y aldagaian gorde). x eta y berdinak diren konprobatu eta mezu bat pantailarazten da.

2.2. Dena ondo joan ezkero, ze balore izan behar ditu argc eta argv­k?

argc = 3

argv[] = ./azterketa, fitx1, fitx2

2.3. Zenbat deskriptore zabaltzen ditu aplikazio honek eta zertarako erabiltzen dira zabaldutakoak?

Bi deskriptore zabaltzen dira. Bietan emandako fitxategiaren inodo datuak zehaztutako struct estrukturan gordetzen dira.

2.4. Zertarako erabiltzen da “perror”?

Sistema deietan errorea gertatzen bada -1 balioa itzultzen dute eta errno aldagaian errorearen zenbakia gordetzen da. perror() metodoaren bidez errno aldagaieko zenbakiari dagokion errore mezua pantailaratuko da.

2.5 Osotu hurrengo hutsuneak:

(1) 3

(2) ./azterketa, sartu datuak horrela: fitx1 fitx2

(3) datuak ezin dira hartu

(4) inodo zenbaki

3. Ariketa

a)

Lehenengoa: Aita

Bigarrena: Semea

b)

if((id=fork())!=0) --> Prozesu berri bat sortuko da, deitzailearen kopia izango dena. Itzulitako balioa id aldagaian gordeko da eta balio hori 0 ez bada (Prozesu umea ez bada, aita bada) baldintzaren barruan sartuko da. (Errorea gertatzen bada ere sartuko da)

for(;;) --> Begizta infinitua.

alarm(5) --> Bost segundo barru SIGALARM seinalea bidaliko zaio prozeduari.

pause() --> Prozesua seinale bat jaso arte egongo da lotan.

kill(id,12) --> Umeari, id, 12 identifikatzailea duen (SIGUSR2) seinalea bidaliko zaio.

c)

sleep(3) --> Prozesu 3 segundu lotan utziko du.

kill(getppid(),10) --> 10 identifikatzailea duen seinalea (SIGUSR1) bidaliko dio bere guraso prozesuari. Gurasoaren identifikatzailea getppid() metodoaren bidez lortzen du.

for(;;) --> Begizta infinitu batean sartuko da.

d)

signal(14, trapper) --> 14 kodea duen seinalea (SIGALARM) trapper funtzioarekin lotzen du. Hau da, prozesu batek seinale hori jasotzean trapper funtzioa exekutatuko du.

signal(10, trapper); --> 10 kodea duen seinalea (SIGUSR1) trapper funtzioarekin lotzen du. Hau da, prozesu batek seinale hori jasotzean trapper funtzioa exekutatuko du.

printf("Prozesu zenbakia: %d\n", getpid()); --> Prozesuaren identifikatzailea pantailaratuko du. Identifikatzailea lortzeko getpid() metodoa erabiltzen du.

signal(sig, trapper); --> Sig identifikatzailea duen seinalea trapper metodoarekin, bere buruarekin, lotzen du. Hau da, prozesu batek seinale hori jasotzean trapper funtzioa exekutatuko du.

e)

Bien exekuzioa aldi berean hasiko da. Aita bost segundu itxaroten egongo da (pause() metodoa exekutatzen dago alarm(5) SIGALARM bidali arte bost segundu geroago) eta ondoren umeari bidaliko dio seinalea (SIGUSR2 da, baina definituta ez dagoenez, defektuz seme prozesua bukatuko du). Semea hiru segundu egongo da lotan (sleep(3)) eta ondoren aitari SIGUSR1 bidaliko dio. Seinale hau trapper metodoarekin dago lotuta, eta horrek egiten duen gauza bakarra jasotako seinalearen zenbakia pantailaraztea da, beraz ez du aita prozesua bukatuko. Hori egin ondoren begizta infinituan sartuko da.

Beraz, lehenengo ume prozesua bukatuko da, baina gurasoak ez du inoiz bukatuko. Hurrengoa pantailaratuko da gutxi gora behera:

Prozesu zenbakia: 6329

Jasotako seinalea 10.

Jasotako seinalea 14.

Jasotako seinalea 14.

Jasotako seinalea 14.

...