Lab1: Signali

1. Općenito o signalima (u UNIX-u)

Signali na razini procesa (pokrenutog programa) su analogni mehanizam sklopovskim *prekidima* na razini procesora. Služe za obradu nekih asinkronih događaja.

Osnovna ideja je kao i kod prekida: proces nešto svoje radi dok se u nekom (neočekivanom) trenutku ne dogodi nešto što treba odmah obraditi. Stoga se pri primitku signala trenutni posao prekida, obrađuje se signal, te po završetku obrade nastavlja s prekinutim poslom.

Signal ne sadrži dodatne informacije (osim broja signala). Signal procesu može poslati jezgra operacijska sustava (npr. SIGSEGV), drugi proces (npr. SIGTERM) ili proces sam sebi (npr. SIGALRM uz odgodu ili neki drugi događaj).

Proces može programirati svoje ponašanje za signal:
1. može ga prihvatiti i obraditi pretpostavljenom funkcijom (većinom to znači prekid procesa)
2. može ga prihvatiti i obraditi funkcijom zadanom u programu (u programu je definirano što će se obaviti po primitku takvog signala)
3. može ga se zadržati (zapamtiti, ali ne i odmah obraditi)
4. može ga se ignorirati (odbaciti bez obrade).

Izuzetak od navedenih pravila je SIGKILL koji ubija proces (SIGKILL ignorira postavke koje je program postavio).

Kada se prihvati signal, pozove se *funkcija za obradu signala*.

Slično kao i kod prekida na razini procesora, i pri prihvatu signala zabranjuje se daljnje prekidanje, ali samo s tim signalom - drugi signali mogu prekinuti tu obradu.

Po završetku obrade ponovno se dozvoljava prekidanje s tim signalom te se eventualno zadržani signali (došli za vrijeme obrade) sada propuštaju.

Signali koji se procesu upute za vrijeme obrade prethodnog signala istog tipa - isti broj, stavljaju se u red i čekaju. S time da se pamti samo po jedan signal istog tipa. Primjerice, ako procesu za vrijeme obrade signala SIGINT dođu još tri takva signala, zapamtit će se samo prvi takav signal. Po završetku obrade prvog signala SIGINT ponovno će se pozvati ista funkcija, ali samo jednom (uz pretpostavku da za vrijeme ove druge obrade nije došao novi signal SIGINT).

Signali su standardni mehanizam na UNIX operacijskim sustavima i njemu sličnim (npr. Linux). Operacijski sustavi Windows imaju minimalnu podršku za signale i u mnogočemu su različiti (zapravo se oni ne koriste).

2. Signali u Pythonu

Osnovna razlika prema "signalima u UNIX-u" jest da se signali u Python programu ne mogu *zadržati* (ponašanje 3. nije izvedivo).

Primjer 1. Za vrijeme obrade signala, novi signali prekidaju obradu novom obradom

```
import signal, time

br_signala = 0

def obrada ( sig, frame ):
	global br_signala
	br_signala = br_signala + 1
	print("Pocetak obrade signala")
	for i in range(10):
		print ( "U obradi signala " + str(sig) )
		time.sleep(0.5)
	print("Kraj obrade signala")

def main():
	print("Signali: obrada se prekida drugim signalima koji se prije obrađuju")

	signal.signal ( signal.SIGINT, obrada )

	for i in range(1,11):
		print("Glavna funkcija: iteracija " + str(i) + " (Ctrl+C za SIGINT)")
		time.sleep(1)

	print ( "Primljeno " + str(br_signala) + " signala za vrijeme rada" )

if __name__ == "__main__":
	main()
```

Primjer 2. signali se ignoriraju za vrijeme obrade (treba postaviti!)

```
import signal
import time

def set_signal_handlers():
	signal.signal(signal.SIGINT, signal_handler)
	signal.signal(signal.SIGTERM, signal_handler)

def ignore_signals():
	signal.signal(signal.SIGINT, signal.SIG_IGN )
	signal.signal(signal.SIGTERM, signal.SIG_IGN )

def signal_handler ( signum, frame ):
	ignore_signals()

	for i in range(1,11):
		print("U obradi signala " + str(signum) + ": iteracija " + str(i) )
		time.sleep(0.5)

	set_signal_handlers()

def main():
	print("Signali: signali koji dođu za vrijeme obrade se ignoriraju")

	set_signal_handlers()

	for i in range(1,11):
		print("Glavna funkcija: iteracija " + str(i) )
		time.sleep(1)

if __name__ == "__main__":
	main()
```

Primjer 3. Programsko rješenje: svaki se signal prihvaća, ali na početku obrade provjerava je li ona već u tijeku

```
import signal
import time

obrada_u_tijeku = 0

def obrada ( signum ):
	print("Pocetak obrade signala")
	for i in range(1,11):
		print("U obradi signala " + str(signum) + ": iteracija " + str(i) )
		time.sleep(0.5)
	print("Kraj obrade signala")

def signal_handler ( signum, frame ):
	global obrada_u_tijeku
	obrada_u_tijeku += 1

	if obrada_u_tijeku > 1: # zapoceta obrada će pozvati i obrade za sve ostale
		return

	while obrada_u_tijeku > 0:
		obrada ( signum )
		obrada_u_tijeku = obrada_u_tijeku - 1

def main():
	print("Signali: obrada se prekida drugim signalima, ali oni se obrađuju "
		"kasnije, nakon prvog (programsko rješenje prihvata)")
	signal.signal(signal.SIGINT, signal_handler)
	for i in range(1,11):
		print("Glavna funkcija: iteracija " + str(i) )
		time.sleep(1)

if __name__ == "__main__":
	main()
```

3. Zadatak za vježbu 1

Simulirati postupak obrade prekida prema prioritetima korištenjem signala. Napisati program *obrada.py* koji omogućava obradu prekida s više razina/prioriteta (simulira ponašanje sustava opisanog u 3. poglavlju i to bez sklopa za prihvat prekida).

Struktura prekidne rutine dana je sljedećim pseudokodom:

```
funkcija prekidna_rutina  /* pokreće se pojavom signala uz zabranu daljih prekida */
	odredi uzrok prekida, tj. indeks i
	OZNAKA_ČEKANJA[i] = 1
	dok je i > TEKUĆI_PRIORITET radi
		OZNAKA_ČEKANJA[i] = 0
		PRIORITET[i] = TEKUĆI_PRIORITET
		TEKUĆI_PRIORITET = i
		omogući prekidanje
		obrada_prekida(i)
		zabrani prekidanje
		TEKUĆI_PRIORITET = PRIORITET[i]
		i = 0
		za j = TEKUĆI_PRIORITET + 1 do N radi
			ako je OZNAKA_ČEKANJA[j]<>0 tada
				i = j
```

Uz pretpostavku da dva signala neće doći jako brzo jedan za drugim, operacije `omogući_prekidanje` i `zabrani_prekidanje` mogu se izostaviti.

Za simulaciju prekida koristiti signal SIGINT koji se može dobiti pritiskom na Ctrl+C.

Predložak i pomoćne funkcije za ispis

```
import time, signal, sys

TEKUCI_PRIORITET = 0
PRIORITET = [0,0,0,0,0,0]
OZNAKA_CEKANJA = [0,0,0,0,0,0]

def ispisi_pojavu_signala ( prioritet ): # prioritet u rangu 1-5
	print ( '\t' + '- ' * prioritet + 'X ' + '- ' * (5-prioritet) )
def ispisi_pocetak_obrade_signala ( prioritet ):
	print ( '\t' + '- ' * prioritet + 'P ' + '- ' * (5-prioritet) )
def ispisi_kraj_obrade_signala ( prioritet ):
	print ( '\t' + '- ' * prioritet + 'K ' + '- ' * (5-prioritet) )
def ispisi_korak_obrade_signala ( prioritet, korak ):
	print ( '\t' + '- ' * prioritet + str(korak) + ' ' + '- ' * (5-prioritet) )
def ispisi_korak_obrade_glavnog_programa ( korak ):
	print ( '\t' + str(korak%10) + ' -' * 5 )

def simulacija_obrade_prekida ( prioritet ):
	ispisi_pocetak_obrade_signala ( prioritet )
	for korak in range(1,6):
		ispisi_korak_obrade_signala ( prioritet, korak )
		time.sleep(1)
	ispisi_kraj_obrade_signala ( prioritet )

def prekidna_rutina ( signal, frame ):
	prioritet = int(input("Unesi prioritet prekida: "))
	if prioritet == 10:
		print ( "\nZavršavam rad" )
		sys.exit(1)

	ispisi_pojavu_signala ( prioritet )
	OZNAKA_CEKANJA[prioritet] = 1;

	# ostatak koda prema predlošku
	# ...
	simulacija_obrade_prekida ( prioritet )
	# ...

def main():
	print("\tG 1 2 3 4 5")

	signal.signal ( signal.SIGINT, prekidna_rutina )

	for korak in range(100):
		ispisi_korak_obrade_glavnog_programa ( korak )
		time.sleep(1)

if __name__ == "__main__":
	main()
```

Primjer pokretanja dovršenog programa

```
        G 1 2 3 4 5
        0 - - - - -
        1 - - - - -
        2 - - - - -
        3 - - - - -
        4 - - - - -

Unesi prioritet prekida: 3
        - - - X - -
        - - - P - -
        - - - 1 - -
        - - - 2 - -

Unesi prioritet prekida: 5
        - - - - - X
        - - - - - P
        - - - - - 1
        - - - - - 2
        - - - - - 3

Unesi prioritet prekida: 2
        - - X - - -
        - - - - - 4
        - - - - - 5
        - - - - - K
        - - - 3 - -
        - - - 4 - -
        - - - 5 - -
        - - - K - -
        - - P - - -
        - - 1 - - -
        - - 2 - - -
        - - 3 - - -
        - - 4 - - -
        - - 5 - - -
        - - K - - -
        5 - - - - -
        6 - - - - -
        7 - - - - -
        8 - - - - -
        9 - - - - -
```