



MODUL PRAKTIKUM SISTEM OPERASI DAN JARINGAN KOMPUTER

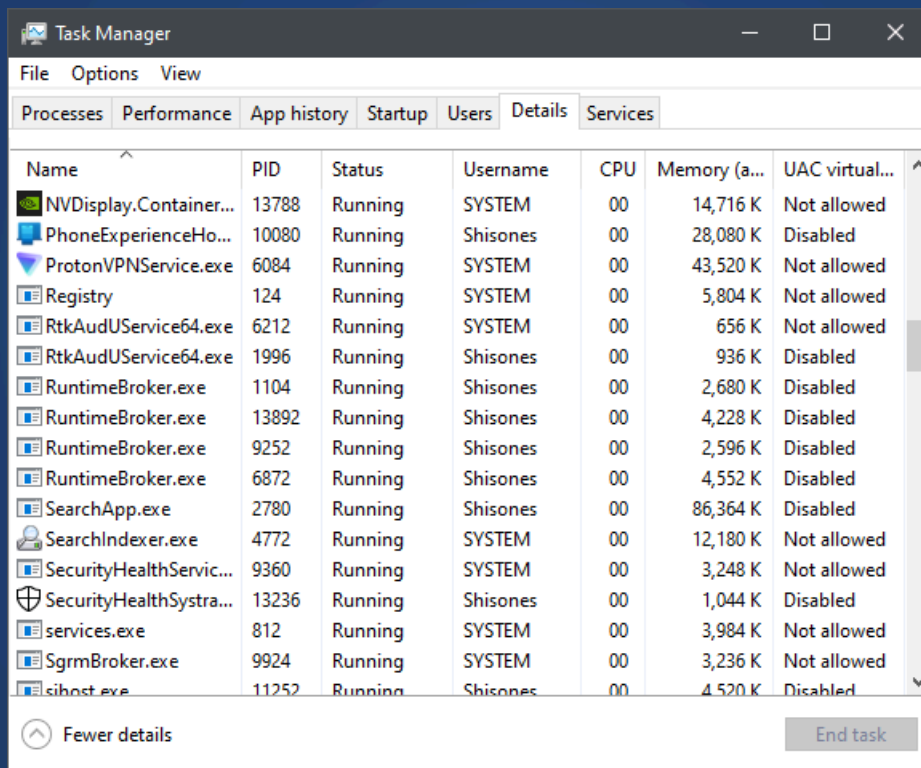
4. Process Management

PROCESSES AND SERVICES

A. Process Management

Process adalah unit dasar eksekusi di dalam sistem operasi Linux. Setiap program yang dijalankan, baik oleh pengguna maupun sistem, adalah proses. Proses melibatkan eksekusi kode dan manajemen memori yang dibutuhkan untuk menjalankan tugas. Proses memiliki identifikasi unik yang disebut **PID (Process ID)** dan berkomunikasi dengan kernel untuk melakukan tugas seperti akses memori, penggunaan CPU, dan interaksi dengan perangkat input/output.

Contoh Proses di Windows:

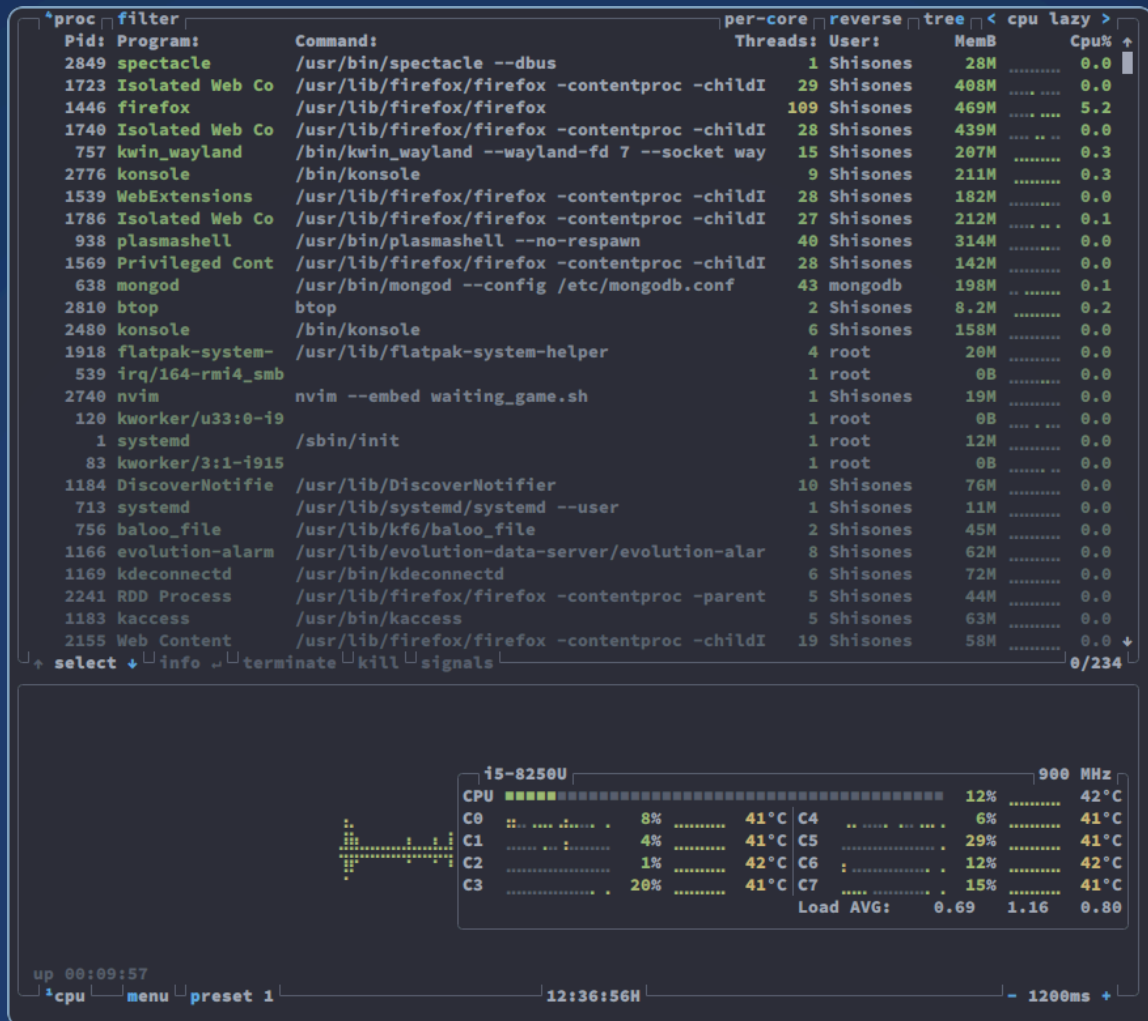


The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running processes with columns for Name, PID, Status, Username, CPU, Memory, and UAC virtualization. The processes listed include NVDIspay.Container..., PhoneExperienceHo..., ProtonVPNService.exe, Registry, RtkAudUService64.exe (multiple instances), RuntimeBroker.exe (multiple instances), SearchApp.exe, SearchIndexer.exe, SecurityHealthServic..., SecurityHealthSystra..., services.exe, SgrmBroker.exe, and sibost.exe.

Name	PID	Status	Username	CPU	Memory (a...	UAC virtual...
NVDIspay.Container...	13788	Running	SYSTEM	00	14,716 K	Not allowed
PhoneExperienceHo...	10080	Running	Shiones	00	28,080 K	Disabled
ProtonVPNService.exe	6084	Running	SYSTEM	00	43,520 K	Not allowed
Registry	124	Running	SYSTEM	00	5,804 K	Not allowed
RtkAudUService64.exe	6212	Running	SYSTEM	00	656 K	Not allowed
RtkAudUService64.exe	1996	Running	Shiones	00	936 K	Disabled
RuntimeBroker.exe	1104	Running	Shiones	00	2,680 K	Disabled
RuntimeBroker.exe	13892	Running	Shiones	00	4,228 K	Disabled
RuntimeBroker.exe	9252	Running	Shiones	00	2,596 K	Disabled
RuntimeBroker.exe	6872	Running	Shiones	00	4,552 K	Disabled
SearchApp.exe	2780	Running	Shiones	00	86,364 K	Disabled
SearchIndexer.exe	4772	Running	SYSTEM	00	12,180 K	Not allowed
SecurityHealthServic...	9360	Running	SYSTEM	00	3,248 K	Not allowed
SecurityHealthSystra...	13236	Running	Shiones	00	1,044 K	Disabled
services.exe	812	Running	SYSTEM	00	3,984 K	Not allowed
SgrmBroker.exe	9924	Running	SYSTEM	00	3,236 K	Not allowed
sibost.exe	11252	Running	Shiones	00	4,520 K	Disabled

Perhatikan bahwa **setiap proses** memiliki **PID yang berbeda**. Masing masing proses memiliki state seperti Running, Ready, Waiting, Suspended, dll.

Di Sistem Operasi Linux, hasilnya sama, hanya beda penamaan saja:



Command `ps` (processes)

```
$ ps aux
```

Menunjukkan seluruh proses yang berjalan.

```
→ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.3	0.0	21056	12544	?	Ss	12:26	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S	12:26	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	12:26	0:00	[pool_workqueue_release]
root	4	0.0	0.0	0	0	?	I<	12:26	0:00	[kworker/R-rcu_gp]
root	5	0.0	0.0	0	0	?	I<	12:26	0:00	[kworker/R-sync_wq]
root	6	0.0	0.0	0	0	?	I<	12:26	0:00	[kworker/R-slub_flushwq]

- **a**: Menampilkan proses dari semua pengguna.
- **u**: Menampilkan informasi tentang usage.
- **x**: Menampilkan proses yang tidak terkait dengan terminal.

Command `top`

```
$ top
```

Seperti task manager di windows, top dapat menunjukkan proses dan usage dari masing-masing proses secara **real time**.

Program program di linux biasanya memiliki versi lebih cantiknya. Contoh untuk top adalah **\$ htop** dan **\$ btop**.

- Top:

```
top - 12:39:10 up 12 min, 2 users, load average: 0.98, 1.11, 0.84
Tasks: 235 total, 2 running, 233 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.9 us, 4.8 sy, 0.0 ni, 81.4 id, 0.0 wa, 0.6 hi, 0.3 si, 0.0 st
MiB Mem : 15884.6 total, 10903.4 free, 3366.2 used, 2753.0 buff/cache
MiB Swap: 5120.0 total, 5120.0 free, 0.0 used, 12518.5 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1446 Shiones  20   0 11.6g 501776 246204 S   51.2   3.1   5:22.09 firefox
 1740 Shiones  20   0 2980176 437768 150276 S   37.6   2.7   2:02.31 Isolated Web Co
  757 Shiones -2   0 2351568 213924 146824 R   17.8   1.3   1:33.36 kwin_wayland
 1723 Shiones  20   0 3019992 418380 117544 S    5.0   2.6   4:45.89 Isolated Web Co
  938 Shiones  20   0 4485168 354876 162900 S    3.6   2.2   0:27.72 plasmashell
 1539 Shiones  20   0 18.8g 188188 105772 S    2.3   1.2   0:32.66 WebExtensions
 1786 Shiones  20   0 2656904 204348 116116 S    2.3   1.3   0:24.77 Isolated Web Co
 2776 Shiones  20   0 1337416 211756 191456 S    2.0   1.3   0:06.25 konsole
  638 mongodb  20   0 3695224 205140 86840 S    1.7   1.3   0:09.59 mongod
 1569 Shiones  20   0 2565800 149600 105880 S    1.7   0.9   0:14.90 Privileged Cont
 1842 Shiones  20   0 2441580 62700 45832 S    1.0   0.4   0:00.74 Web Content
  209 root       20   0      0      0      0 I    0.7   0.0   0:00.32 kworker/1:2-events
  539 root      -51  0      0      0      0 S    0.7   0.0   0:05.07 irq/164-rmi4_smbus
 3038 Shiones  20   0 9692   7004 4956 R    0.7   0.0   0:00.56 top
  46 root       20   0      0      0      0 S    0.3   0.0   0:00.14 ksoftirqd/4
  72 root       20   0      0      0      0 I    0.3   0.0   0:00.59 kworker/u32:2-events_unbound
  81 root      -51  0      0      0      0 S    0.3   0.0   0:00.32 irq/9-acpi
  83 root       20   0      0      0      0 I    0.3   0.0   0:02.49 kworker/3:1-i915-unordered
```

- HTop:

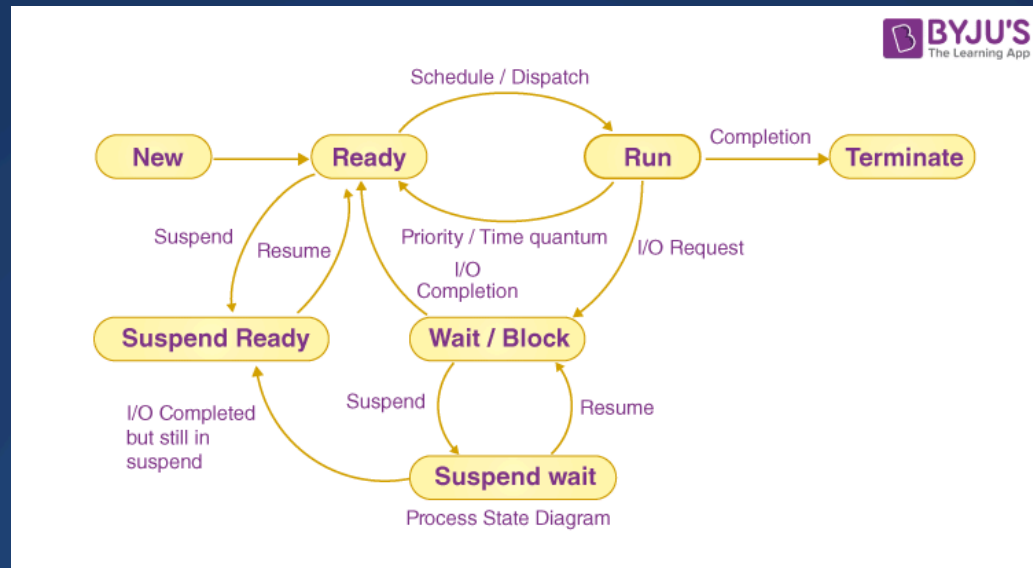
```
0[||||| 4.7%] 4[||||| 6.4%]
1[||||| 9.2%] 5[ 0.0%]
2[||| 1.7%] 6[||| 2.9%]
3[| 0.6%] 7[||||| 25.7%]
Mem[||||| 2.98G/15.5G] Tasks: 86, 608 thr, 150 kthr; 3 running
Swp[ 0K/5.00G] Load average: 0.84 1.05 0.83
Uptime: 00:12:59

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 3180 Shiones  20   0 9716 7464 5288 R   11.0   0.0   0:01.96 htop
  757 Shiones -2   0 2249M 210M 144M S    1.7   1.3   1:28.89 /bin/kwin_wayland --wayland-fd 7 --socket wayland-0
  822 Shiones -2   0 2249M 210M 144M S    0.0   1.3   0:06.71 /bin/kwin_wayland --wayland-fd 7 --socket wayland-0
 2776 Shiones  20   0 1274M 207M 187M S    1.2   1.3   0:07.82 /bin/konsole
  658 mongodb  20   0 3608M 200M 86840 S    0.0   1.3   0:00.16 /usr/bin/mongod --config /etc/mongodb.conf

  938 Shiones  20   0 4363M 353M 159M S    0.0   2.2   0:13.37 /usr/bin/plasmashell --no-respawn
  949 Shiones  20   0 4363M 353M 159M S    0.0   2.2   0:01.28 /usr/bin/plasmashell --no-respawn
  987 Shiones  20   0 976M 62824 50464 S    0.0   0.4   0:00.32 /usr/lib/xdg-desktop-portal-kde
 1020 Shiones  20   0 976M 62824 50464 S    0.0   0.4   0:00.05 /usr/lib/xdg-desktop-portal-kde
 1740 Shiones  20   0 2914M 431M 149M S    0.0   2.7   1:45.59 /usr/lib/firefox/firefox -contentproc -childID 5 -i
 1786 Shiones  20   0 2594M 200M 113M S    0.0   1.3   0:18.06 /usr/lib/firefox/firefox -contentproc -childID 6 -i
 2778 Shiones  20   0 1274M 207M 187M S    0.0   1.3   0:00.24 /bin/konsole
    1 root       20   0 21056 12544 9044 S    0.6   0.1   0:01.99 /sbin/init
  318 root       20   0 47608 17352 16456 S    0.6   0.1   0:00.45 /usr/lib/systemd/systemd-journald
  336 root       20   0 14288 5832 5064 S    0.0   0.0   0:00.03 /usr/lib/systemd/systemd-userdbd
  365 root       20   0 34620 10368 7424 S    0.0   0.1   0:00.26 /usr/lib/systemd/systemd-udev
  493 dbus       20   0 7356 3732 2948 S    0.0   0.0   0:00.07 /usr/bin/dbus-broker-launch --scope system --audit
  494 dbus       20   0 6084 3712 2092 S    0.0   0.0   0:00.51 dbus-broker --log 4 --controller 9 --machine-id c11
  495 root       20   0 403M 24628 20916 S    0.0   0.2   0:00.43 /usr/bin/NetworkManager --no-daemon
  496 root       20   0 9704 5248 4864 S    0.0   0.0   0:00.09 /usr/lib/bluetooth/bluetoothd
  497 root       20   0 300M 6364 5852 S    0.0   0.0   0:00.06 /usr/lib/boltld
  498 root       20   0 15144 7880 6760 S    0.0   0.0   0:00.29 /usr/lib/systemd/systemd-logind
```

1. Process State

Masing-masing Proses memiliki state-nya sendiri, contohnya seperti ini



- **New**: Proses baru dibuat, tetapi belum di-load ke memori utama.
- **Suspended Wait**: Proses di pause dan menunggu sumber daya eksternal.
- **Suspended Ready**: Proses di pause tetapi siap dijalankan ketika dipindahkan kembali ke memori.
- **Waiting**: Proses menunggu event atau sumber daya untuk melanjutkan eksekusi.
- **Ready**: Proses siap dijalankan dan menunggu giliran di CPU.
- **Run**: Proses sedang aktif berjalan di CPU.
- **Terminated**: Proses telah selesai dan dihentikan dari eksekusi.

2. Kill Signals

Sinyal (signals) adalah mekanisme interupsi yang memungkinkan proses di Linux untuk menerima instruksi tertentu dari sistem atau pengguna. Beberapa sinyal yang umum digunakan adalah:

SIGTERM (Sinyal 15): Meminta proses untuk berhenti secara normal.

SIGKILL (Sinyal 9): Menghentikan proses secara paksa.

SIGSTOP (Sinyal 19): Menghentikan (pause) proses tanpa mengakhiri

SIGCONT (Sinyal 18): Melanjutkan proses yang dihentikan SIGSTOP.

SIGINT (Sinyal 2): Menghentikan proses yang berjalan di foreground (biasanya dikirim dengan Ctrl+C).

Sinyal ini digunakan oleh command **\$ kill**, dimana command tersebut dapat mengubah state dari suatu proses.

Contoh :

- Jalankan proses nano
- Cari proses nano menggunakan terminal baru
- Lakukan command dibawah ini

```
$ kill -9 [PID dari nano]
```

Close Proses nano dari terminal lain (sama seperti close paksa dari task manager)

3. Foreground and Background Jobs

Di Linux, proses dapat dijalankan di **foreground** atau **background**. Saat Anda bekerja di terminal, biasanya proses yang dijalankan akan langsung berjalan di **foreground** (proses yang aktif di terminal dan menerima input/output langsung dari pengguna). Namun, Anda juga dapat menghentikan proses sementara atau menjalankannya di background.

Ketika Anda menjalankan suatu proses di **foreground**, seperti editor teks atau aplikasi lain, Anda bisa menghentikan proses tersebut sementara (suspend) dengan menekan **Ctrl+Z**. Ini akan mengirimkan sinyal **SIGSTOP** yang menghentikan proses dan menempatkannya di daftar **background jobs** sebagai proses yang dihentikan.

Contoh:

- Buka suatu file dengan nano
- Ctrl+Z

[1]+ Stopped nano file.txt

^ ini artinya proses nano tadi distop

Setelah proses dihentikan, Anda memiliki dua opsi untuk melanjutkan proses tersebut:

- **fg (foreground)**: Melanjutkan proses di foreground, artinya proses akan kembali aktif di terminal dan mengambil alih terminal.

```
$ fg %1
```

- **bg (background)**: Melanjutkan proses di background, artinya proses akan berjalan di belakang layar tanpa mengambil alih terminal, sehingga terminal tetap bisa digunakan untuk tugas lain.

```
$ bg %1
```

Untuk melihat proses fg/bg yang berjalan, kita bisa lakukan command

```
$ jobs
```

Kita dapat menjalankan suatu command di terminal sekaligus mengirimnya ke background dengan

```
$ nano file.txt &
```

4. Prioritas Proses

Linux menggunakan sistem multitasking di mana beberapa proses bisa berjalan bersamaan. Ketika beberapa proses membutuhkan CPU, Linux menggunakan prioritas proses untuk **menentukan proses mana yang mendapatkan lebih banyak waktu CPU**. Salah satu cara untuk mempengaruhi prioritas ini adalah dengan menggunakan **nice value**.

Nice Value adalah angka yang digunakan untuk mengatur prioritas relatif dari suatu proses terhadap proses lain. Semakin tinggi nilai nice, **semakin "baik hati" (nice)** proses tersebut terhadap proses lain, artinya ia akan mendapat **prioritas lebih rendah** untuk menggunakan CPU.

Nilai nice berkisar dari -20 (prioritas tertinggi) hingga 19 (prioritas terendah). Proses dengan nilai nice yang lebih rendah (misalnya, -10) akan mendapat lebih banyak waktu CPU dibandingkan proses dengan nilai nice yang lebih tinggi (misalnya, 10). ***nilai nice negatif hanya untuk root**

Nice Value	Prioritas CPU
-20	Prioritas tertinggi
0	Prioritas default (normal)
19	Prioritas terendah

```
$ nice -n 10 sleep 1000
```

^ set nice value ke 10 untuk command \$ sleep 1000

```
$ ps -o pid,ni,cmd -p $(pgrep sleep)
```

Untuk melihat nice value dari sleep

```
renice 5 -p <PID>
```

Untuk mengeset ulang nice value dari suatu proses

Kesimpulan

\$ nice digunakan untuk mengatur prioritas proses.

\$ renice digunakan untuk mengubah prioritas proses yang sudah berjalan.

B. Tugas Praktikum

Anda adalah seorang administrator sistem di sebuah perusahaan pengembangan perangkat lunak. Tim Anda sedang melakukan pengujian performa aplikasi baru yang bernama **waiting_game.sh**. Aplikasi ini dirancang untuk menghabiskan waktu CPU dan RAM dengan cara menunggu. Tim pengembang meminta Anda untuk membantu mereka memantau penggunaan sumber daya aplikasi ini dan memahami bagaimana prioritas proses (**nice value**) mempengaruhi performa aplikasi.

1. Monitoring Proses Menggunakan **top** atau **htop**

Jalankan perintah **top** atau **htop** di terminal untuk memantau semua proses yang berjalan di sistem. Ambil **screenshot yang menunjukkan penggunaan CPU dan memori** sebelum menjalankan aplikasi **waiting_game.sh**.

2. Suspended Process

Setelah aplikasi **waiting_game.sh** berjalan, tekan **Ctrl+Z** untuk menghentikannya sementara. Catat status proses yang dihentikan di terminal. Kemudian **jalankan waiting_game.sh di Background dan Foreground**.

3. Setup Priority

Buka dua terminal berbeda. Di terminal pertama, jalankan aplikasi **waiting_game.sh** dengan **nice value 19** (prioritas rendah), Di terminal kedua, jalankan aplikasi yang sama dengan **nice value 1** (prioritas tinggi): Bandingkan dan catat performa dari masing-masing tes. Amati penggunaan **CPU dan memori** pada kedua terminal.

*yang dikumpulkan hanya pdf laporan ges



```
#!/bin/bash
```

```
# Loop for 60 seconds
for i in {1..60}
do
    # Output the current time
    echo $(date +"%H:%M:%S")
    for j in {1..$i*10}
    do
        echo "
    done

    # Wait for 1 second
    sleep 1
done
```

^ isi dari waiting_game.sh

